Message Authentication and Recognition Protocols Using Two-Channel Cryptography

by

Atefeh Mashatan

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Combinatorics & Optimization

Waterloo, Ontario, Canada, 2008

Abstract

We propose a formal model for non-interactive message authentication protocols (NIMAPs) using two channels and analyze all the attacks that can occur in this model. Further, we introduce the notion of hybrid-collision resistant (HCR) hash functions. This leads to a new proposal for a NIMAP based on HCR hash functions. This protocol is as efficient as the best previous NIMAP while having a very simple structure and not requiring any long strings to be authenticated ahead of time.

We investigate interactive message authentication protocols (IMAPs) and propose a new IMAP, based on the existence of interactive-collision resistant (ICR) hash functions, a new notion of hash function security. The efficient and easy-to-use structure of our IMAP makes it very practical in real world ad hoc network scenarios.

We also look at message recognition protocols (MRPs) and prove that there is a one-to-one correspondence between non-interactive MRPs and digital signature schemes with message recovery. Further, we look at an existing recognition protocol and point out its inability to recover in case of a specific adversarial disruption. We improve this protocol by suggesting a variant which is equipped with a resynchronization process. Moreover, another variant of the protocol is proposed which self-recovers in case of an intrusion. Finally, we propose a new design for message recognition in ad hoc networks which does not make use of hash chains. This new design uses random passwords that are being refreshed in each session, as opposed to precomputed elements of a hash chain.

Acknowledgments

I accomplish nothing without the grace of the Creator, the devotion from my teachers, and the support of my family and friends.

I would like to offer my deepest appreciations to Professor Douglas Stinson who gave me the honour of being his PhD student. Words cannot explain the extent of my gratitude for his invaluable support. I was very fortunate that he opened his office to me and gave me the chance to work with him when other doors were closed. The positive flow of energy present in his office transforms all the frustrations to hope, optimism, and determination; all of which was definitely vital for this thesis.

I gratefully acknowledge Professor Edlyn Teske for her advice and guidelines. I would like to especially thank her for her supervision during the first two years of my PhD. I also benefited much from her constructive comments and encouragements while I was writing my thesis.

It is a great pleasure to thank Professor Alfred Menezes for his help and support. I made several visits to his office while I was a PhD student and discussed research problems or asked his opinion about a decision I had to make. His advice has nourished my intellectual maturity which I will continue to benefit from.

I would like to express my profound gratitude to Dr. Ian Goldberg for his constructive criticism that improved this thesis to a great extent. I had the pleasure of writing a joint paper with him in the course of writing this thesis. I am grateful in every possible way for his contribution to this thesis and hope for future collaboration with him.

My light during the dark nights, my guide through the difficult journeys, and my first teachers in life, *maman* and *baba*, Fereshteh Izadian and Hossein Mashatan, how would I ever accomplish anything without your unconditional love and indefinite support? I am as ever, highly indebted to you pillars of my being. I would like to acknowledge my younger brothers, Vahid and Farid, for their love and support, and of course their little distractions which are cute and charming!

Table of Contents

Chapter 1

Introduction

Contents

In this thesis, we focus on using two-channel cryptography to design message authentication protocols and message recognition protocols suitable for networks consisting of devices with limited resources. In particular, we look at non-interactive message authentication protocols (NIMAPs), interactive message authentication protocols (IMAPs), and message recognition protocols (MRPs). Previous protocols are reviewed and some are improved; new protocols are proposed; security proofs are provided; and advantages of the new protocols are highlighted.

Standard models of public-key cryptography and secret-key cryptography have addressed the problem of message authentication and message recognition by means of assuming availability of public-key infrastructures or secure channels. In some scenarios, however, assuming the traditional settings of public-key and secret-key cryptography might not be practical and, indeed, using these techniques may be very costly. Mobile ad hoc networks (MANET), wireless sensor networks (WSN), and pervasive networks in general are examples of scenarios in which traditional cryptographic protocols may not be suitable, or not even possible, to implement.

In search of a solution to this problem, researchers realized that when the devices come in close geographic proximity of each other, it is possible to make use of a *manual channel*, as well as the usual wireless channel. Instances of the manual channel are typically more expensive to operate compared to the wireless channel. However, they provide some level of security. For example, the channel may provide authenticity of short messages, but may not be confidential. The aim is to employ a (broadband and insecure) wireless channel and a (somewhat secure and narrow-band) manual channel at the same time and attain a security objective, message authentication for instance. This motivated the term *two-channel cryptography*.

In 1984, Rivest and Shamir [RS84] first proposed incorporating human participation in authentication protocols. However, this idea did not receive serious attention from researchers until very recently. Rivest and Shamir proposed using the human voice in authentication protocols and they designed a protocol for two parties who want to authenticate a key under the assumption that no trusted third parties exist, but where the two parties can recognize each other's voices. They motivate the setting by the following scenario. Two company executives who can recognize each other's voices but who do not have each other's public keys want to exchange keys via a scrambled telephone line. They begin by exchanging their public keys. Then, each user chooses a message and uses the other party's public key to encrypt the message. The first half of the bits of the resulting ciphertexts are exchanged. The parties acknowledge receiving the first half of the ciphertexts on the phone. Then, the second half of the bits are sent. While all the steps of the protocol are handled automatically, the two executives are aware of each other's unscrambled voice and use this knowledge to verify the authenticity of the strings.

1.1 Two-channel Cryptography and Applications

We first describe the communication model of two-channel cryptography, where it is assumed that two channels are accessible for communication: an *insecure broadband channel*, denoted by "$\rightarrow$", and an *authenticated narrow-band channel*, denoted by "$\Rightarrow$". Communication over the authenticated channel is usually more expensive and less convenient. Hence, the messages sent over the authenticated channel are usually much shorter than those sent over the insecure channel. The goal of two-channel cryptography is, then, to achieve a certain cryptographic objective by means of the two channels, while optimizing the cost.

In some scenarios, the narrow-band authenticated channel may be accessible all the time, whereas in other scenarios, it may accessible only during the initialization phase. An insecure wireless channel is an example of the broadband channel. The narrow-band channel is usually used to send a short string. Instances of the narrow-band channel include voice-over-internet-protocol (VoIP), data imprinting or data comparison by a user, near field communication (NFC), infrared (IR), laser, or visible light between two devices.

The assumed adversarial capabilities may vary depending on the particular scenario. However, the following are common assumptions on what an adversary can and cannot do in two-channel cryptography.

- The adversary has full control over the broadband channel. That is, the adversary can listen to any messages sent over the broadband channel, modify the messages sent via this channel, stall a message from being delivered, and insert a new message into this channel at any time.
- On the other hand, we assume that the adversary's control over the authenticated channel is limited. In particular, the adversary cannot modify the information transmitted over the authenticated channel, i.e., data integrity is ensured in this channel. However, it may be possible to read, delay or remove a message from this channel.

Moreover, the authenticated channel is equipped with user authenticating features such that the recipient of the information can be sure about who sent it. In other words, an adversary cannot initiate a flow over this channel. On the other hand, the adversary, for instance in case of human VoIP, may be able to replay a previous flow sent through this channel. However, replaying a previous flow sent by Alice to Bob is not going to help Eve, when she wants to deceive another party, Charlie. That is, when Bob receives an authenticated flow, he can check if he was the intended recipient or not. In other realizations of the narrow-band channel, however, replaying an authenticated flow may not be possible.

Two-channel cryptography techniques have several applications, especially in constrained environments where secure channels or trusted infrastructures do not exist or are very costly to provide. Moreover, these techniques are useful in networks that are composed of constrained devices which cannot handle heavy computations such as public-key computations.

With new technological advancements in miniaturizing devices and the emerging smart homes and buildings projects [CEEC08], the problem of designing lightweight cryptographic protocols for low-end devices has attracted a lot of attention both in the academic community and in industry. In scenarios such as personal area networks (PAN) [GN04] and telemedicine (remote health care where medical personnel can monitor the patients from a distance) [Dem04], where the devices are naturally attended by users, the idea of employing the manual channel is even more appealing. This approach is especially attractive when it enables researchers to design more cost-efficient and easy-to-implement protocols.

Another important application is disaster recovery, when a trusted infrastructure is compromised. The use of two-channel cryptography allows for temporary, yet speedy, relief before the infrastructure is fully recovered. Full recovery usually takes a lot longer and security providers need to be vigilant in the meantime.

1.2 Message Authentication in Ad hoc Networks

The problem of authentication is an important aspect of secure communication. Typically, communicating parties would like to be assured of the authenticity of information they obtain via potentially insecure channels.

An ad hoc network is a network where some of the users are part of the network only for a short period of time. For practical reasons, it should be possible to quickly add new users to an ad hoc network. In this network, like any other network, it is desirable to have message authentication. However, assuming traditional settings might not be practical. For example, a public-key infrastructure may not exist; secure channels might not be present; communication bandwidth may be severely limited. Consider the following scenario presented in the literature [BSSW02] which motives this setting: a traveller in an airport lounge would like to print a sensitive document from his or her laptop to one of the many printers set up in the airport lounge. The lounge does not have a secure universal naming infrastructure for the printers. The traveller wants to choose a particular printer and make sure the document gets printed by *that* particular printer (and no other printer), using the insecure wireless channel. The traveller's laptop and a printer need to be securely introduced while there is no public-key infrastructure or secure channel available.

In order to overcome these difficulties in an ad hoc network and still be able to provide message authentication, one can employ two-channel cryptographic techniques when designing protocols [BSSW02, GN04, GMN04, Hoe04, LN06, LAN05, NSS06, PV06, RWSN07, SA99, Vau05, WSN08].

We focus on message authentication protocols which deploy both narrow-band and broadband channels between a claimant Alice and a verifier Bob. Alice chooses a message $M \in \mathcal{M}$, where $\mathcal{M}$ denotes the space of all acceptable messages, and sends it to Bob using a NIMAP or an IMAP. At the end of the protocol, Bob either outputs (Alice, M'), where $M' \in \mathcal{M}$, or he rejects. In the absence of an active adversary, denoted as Eve, the message M sent from Alice should be recovered by Bob, making him accept and output (Alice, M). This message M could be a key that is going to be used for further communication. Eve's goal is to make Bob accept a message M' along with the identity of Alice, when Alice has never sent M'.

The attack model assumed in this context is the adaptive chosen plaintext attack (ACPA) model [GMR88]. The ACPA model consists of two phases: an *information gathering stage* and a *deception* stage. In the information gathering stage, Eve adaptively makes Alice send $M_1, M_2, \ldots, M_q$ to Bob, where q is an integer termed the *querying complexity.* In the deception stage, Eve sends a single message M', along with the identity of Alice, to Bob, where $M' \notin \{M_1, M_2, \ldots, M_q\}$. Eve is successful if Bob accepts M' along with Alice's identity. The computational complexity of the adversary before the deception starts, i.e. during the information gathering stage, is referred to as *offline computational complexity*, whereas *online computational complexity* refers to the computational complexity of the adversary during the deception stage.

1.3 Message Recognition in Ad Hoc Networks

Message recognition in ad hoc networks has been motivated in the literature by the following example [LZWW05]. Consider Alice and Bob, two strangers who meet at a party for the first time. They make a bet before they leave the party. Later, the outcome turns out to be in favour of Alice, and a few days later, Bob receives a message claiming to be sent from Alice. The message includes a bank account number and asks Bob to deposit Alice's prize to that bank account. How can Bob be assured that this message was indeed sent from the entity who introduced herself as "Alice" in the party? That is, Bob wants to *recognize* "Alice", whoever she was, or a message that was sent from her. This problem has a solution if Alice and Bob exchange some information, which is not necessarily secret, at the party.

Alternatively, let Alice and Bob be two small devices in a hostile environment. They have previously "met" in an environment that allowed them to send authenticated messages, but the messages were not confidential. Later, Alice wants Bob to recognize her or recognize the messages sent from her to Bob. There is an adversary, Eve, who is trying to make Bob recognize Eve as Alice, or accept messages from Eve as sent from Alice, where Alice has never sent those messages. Note that we do not consider replay attacks as threats. A message recognition protocol is considered to be secure if Eve's attempts are detected by Alice or Bob.

Recall the following widely used definitions of entity authentication and message authentication from the Handbook of Applied Cryptography [MvOV96]. Entity authentication is a security notion which assures the identity of a participating party to a second party. Message authentication, on the other hand, provides data origin authentication with respect to the original message source and does not have to provide uniqueness and timeliness.

We now define *entity recognition*, a security notion related to the entity authentication. Entity recognition is a weaker security notion than entity authentication; entity recognition refers to the process where two parties meet initially and one party can be assured in future conversations that it is communicating with the same second party. It should also provide *uniqueness*, that is, the corroborative evidence obtained in this process should uniquely correspond to the identity of the claimant. It should also assure *timeliness*, that is, to provide verifiable evidence that the claimant is active at the time of, or immediately before, the evidence was obtained.

Message recognition is a weaker security notion than message authentication and it provides data integrity with respect to the data origin. It ensures that the entity who sent the message is the same in future conversations. However, it does not have to provide uniqueness or timeliness.

Public-key techniques such as digital signature schemes solve the problem of recognition easily. However, using these techniques in some scenarios may be very costly. For instance, there may be no pre-deployed authentic information accessible. Also, we may not be able to assume trusted third parties are available to form a trusted infrastructure. Further, we may be dealing with devices with very low computational power where public-key computations are too heavy to be carried out. On the other hand, secret-key techniques require the existence of a secure channel where the secret keys can be transmitted confidentially. In a dynamic environment with no infrastructure, this assumption may not be easily realized.

The question is which security objectives can be achieved among devices with low-computational power in an environment where no pre-established authentic information exists and without the presence of a trusted third party. Weimerskirch and Westhoff [WW03] argued that in such an environment, achieving authentication is not possible and that all one can achieve is recognition security. Further, it is noted [WW03] that recognition is often all that is required in most dynamic environments. Hence, the weaker security goals of entity and message recognition are often pursued [ABC$^+$98, HWGW05, LZWW07, LZWW08, Mit03, WW03].

There are two communication channels considered in the setting of recognition protocols: an insecure broadband channel which is available all the time, and an authenticated non-confidential narrow-band channel, which is only accessible once, at the very beginning of the protocol. That is, the narrow-band channel is used for the initial session between two users and later sessions occur over the insecure channel.

1.4 Interactive versus Non-interactive Protocols

A message authentication protocol may or may not require online interaction with Bob. There are non-interactive and interactive message authentication protocols that have been considered in the literature [BSSW02, GN04, GMN04, Hoe04, LN06, LAN05, NSS06, PV06, RWSN07, SA99, Vau05, WSN08].

In a NIMAP, all flows are initiated by Alice. She sends some information over the broadband channel and some information over the narrow-band channel. Since there is no flow being initiated by Bob, the order in which Alice's flows are sent is irrelevant. As a result, we can combine all flows sent over the broadband channel into one single flow and, similarly, we can combine all flows sent over the narrow-band channel into one single flow. Hence, without loss of generality, we obtain a typical flow structure of a NIMAP as depicted in Fig. 1.1.

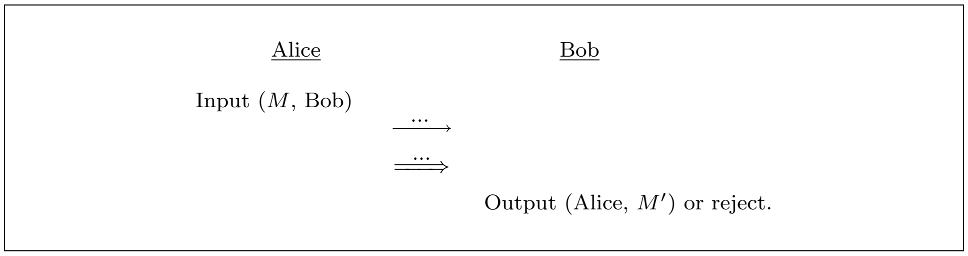


Figure 1.1: A Schematic NIMAP

On the other hand, the flow structure of an IMAP can be more complicated. There is at least one flow initiated by Bob and, hence, the order in which flows are initiated matters. There may be more than one narrow-band flow. The authenticated channel may be bidirectional which means Bob can initiate a flow over the narrow-band channel as well. Illustrated in Fig. 1.2 is a possible flow structure of an IMAP. In this particular flow structure, the first flow is initiated by Alice on the broadband channel which is followed by a response from Bob on the same channel. Then, Alice sends one more flow over the broadband channel and her authenticated flow over the narrow-band channel.

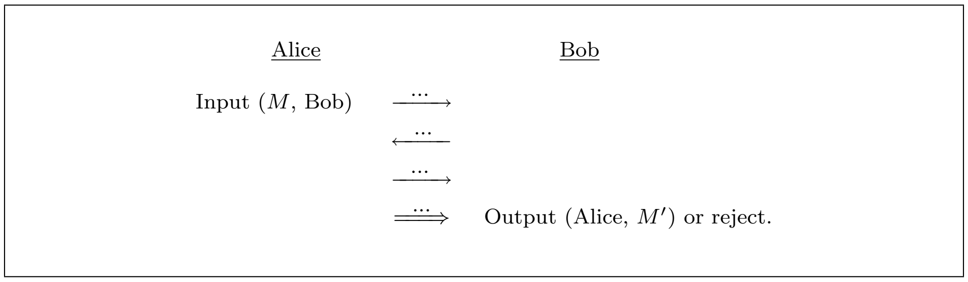


Figure 1.2: A Sample Schematic IMAP

NIMAPs are particularly interesting because they do not require the verifier to be online. On the other hand, interaction sometimes allows for more efficient protocols. Furthermore, some objectives may not be achievable in the non-interactive setting, but can be realized in an interactive setting.

1.5 Computational versus Unconditional Security

In the unconditional security setting, the adversary is assumed to have unlimited computational resources. In the computational security setting, on the other hand, the computational power of the adversary is bounded (typically, it is assumed to be polynomial-time, as a function of a certain security parameter). Moreover, the querying complexity of the adversary is also bounded in the computational security settings. In order for a protocol to be considered secure, the best currently-known methods to defeat a system or protocol should exceed the computational resources of the adversary by a comfortable margin. In case of computationally secure NIMAPs, IMAPs, or MRPs, a successful adversary is reduced (in the sense of a Turing reduction) to an attacker against a well-known system or problem which is proven, or widely believed, to be secure.

1.6 Contributions of this Thesis

There have been many recent papers written on the topic of authentication [BSSW02, GN04, GMN04, Hoe04, LN06, LAN05, NSS06, PV06, RWSN07, SA99, Vau05, WSN08] or recognition [ABC+98, HWGW05, LZWW05, LZWW07, LZWW08, Mit03, WW03] for low-end devices in constrained environments where a low-bandwidth authenticated channel is accessible. We analyze all the existing protocols in this context and point out some of their shortcomings. We also improve some existing protocols, for example by proposing new and more efficient protocols which are based on fewer security assumptions, and by providing a general framework, where possible, to enable a more unified approach to analyzing such protocols.

In the first part of this dissertation, we examine the topic of message authentication protocols in ad hoc networks. Previous interactive and non-interactive protocols from the literature are fully analyzed. In Chapter 2, we describe a formal model for NIMAPs using two channels, and analyze the attacks that can occur in this model. The attack model considered is strong and a scheme that is proved secure in this model does not require authenticated channels that have any unusual properties. Further, the idea of hybrid-collision resistant (HCR) hash functions is introduced and analyzed. This leads to a new proposal [MS07] for a NIMAP based on HCR hash functions. This protocol is considered in the computational security setting and it is as efficient as the best previous computationally secure NIMAP while having a very simple structure and not requiring any long strings to be authenticated ahead of time. Finally, we provide a new proof of non-existence of nontrivial unconditionally secure NIMAPs. This proof consists of a combinatorial counting argument and is much shorter than the previous proof [WSN08].

In Chapter 3, we investigate IMAPs and present a new computationally secure IMAP [MS08a], based on the existence of interactive-collision resistant (ICR) hash functions, a new notion of hash function security. The security of this IMAP is based on the computational assumption that ICR hash functions exist. It performs better than other message authentication protocols that are based on computational assumptions. That is, while achieving the same level of security, the amount of information sent over the authenticated channel in our IMAP is smaller. The efficient and easy-to-use structure of our IMAP makes it very practical in real world ad hoc network scenarios. Finally, we propose a generalization of an unconditionally secure IMAP. We give sufficient conditions for such an IMAP to be secure.

The second part of the dissertation is devoted to examining the problem of message recognition. In Chapter 4, we prove that there is a one-to-one correspondence between non-interactive MRPs and digital signature schemes with message recovery. Further, we improve the best existing recognition protocol due to Lucks et al. [LZWW05] by suggesting a variant [MS08c] to overcome a certain shortcoming. In particular, in case of communication failure or adversarial disruption, the

Lucks et al. protocol was not equipped with a practical resynchronization process and therefore it could fail to resume. We propose a new variant of this protocol [MS08c] that is equipped with a separate resynchronization technique that allows users to resynchronize whenever they wish or when they suspect an intrusion has taken place. Further, we present another variant of the protocol [GMS08], which "self-recovers" in case of an intrusion; it does not need a separate resynchronization process.

Previous recognition proposals in the literature were based on the idea of exchanging values of a hash chain [HWGW05, LZWW05, WW03]. In particular, each pair of users wishing to communicate required a separate pair of hash chains, which puts a relatively heavy memory requirement on low-end devices such as sensor motes, (nodes in a wireless sensor network). Furthermore, the security assumptions for this protocol depend on the number of sessions the protocol has been executed, which gives rise to some undesirable security constraints.

We propose a new design for message recognition [MS08b] in ad hoc networks and explain the advantages of using this new design as compared to previous alternatives. Our proposed recognition protocol does not make use of hash chains. Instead, the keys used in this protocol are chosen at random in each session. As a result, we no longer require the low-end devices to save values of a hash chain in their memories, thus relaxing the memory requirements. Moreover, the keys are independent of one another and are refreshed in each session. This can be done an arbitrary number of times, so we do not need to fix the total number of times the protocol can be executed. As the passwords corresponding to each session are chosen at random and are independent of one another, we do not need to consider assumptions that depend on the number of sessions the protocol is executed. Consequently, the security does not weaken as the protocol is executed repeatedly over time. Last but not least, we provide a practical procedure for resynchronization in case of any possible adversarial disruption or communication failure.

Chapter 2

Non-interactive Message Authentication Protocols

Contents

In this chapter, we consider the problem of non-interactive message authentication using two-channel cryptography: an insecure broadband channel and an authenticated narrow-band channel. This problem has been considered in the context of ad hoc networks, where it is assumed that there is neither a secret key shared among the two parties, nor a public-key infrastructure in place. The model we consider is described in detail in the literature [GN04, GMN04]. Two small devices wish to establish a secure key in an environment where no public-key infrastructure exists. The two devices can communicate over an insecure broadband network. Also available is an authenticated narrow-band channel. This channel might be based on information transmitted by human beings, e.g., a short string that is read from one device and copied to the other device. The short string is used to help to authenticate the information sent over the wide-band channel.

In Section 2.1, we present a formal framework [MS07] for protocols of this type, termed as General Non-Interactive Message Authentication Protocol (GNIMAP). When discussing the security of GNIMAP, we prove that given that a Binding Game is hard to win for an adversary with certain properties, GNIMAP is computationally secure. One can use this framework to analyze and compare particular instances of a Non-Interactive Message Authentication Protocol (NIMAP) in a more unified and consistent approach.

We continue in Section 2.2 by briefly examining the previous NIMAPs available in the literature. We look at the performance and security of these protocols with respect to our general framework.

This chapter is continued by presenting a new NIMAP [MS07] which is as efficient as the best previous NIMAPs, while it benefits from a simpler and easier to implement structure. The security of our protocol is based on a new property of hash functions that we introduce, which we name Hybrid-Collision Resistance (HCR). We analyze the HCR notion in the random oracle model to compare it with more standard notions of hash function security.

This chapter is concluded by proving that nontrivial unconditionally secure NIMAPs do not exist. Wang and Safavi-Naini [WSN08] first proved this nonexistence result using probability distribution arguments. We prove the same result using a simple counting argument which is much shorter.

2.1 General Framework: GNIMAP

We consider a non-interactive Message Authentication Protocol that employs both the authenticated and the insecure channel between a claimant Alice and a verifier Bob. All flows are initiated from Alice and there are a total of two flows, one over the insecure channel and the other over the authenticated channel. We note that there is no flow being initiated from Bob and as a result, the order in which these two flows are being sent over the channels does not matter. Moreover, all other scenarios of a non-interactive Message Authentication Protocol involving more than two flows can be reduced to this scenario. That is, we can simply combine the flows sent over each type of channel in a single flow. This is not the case in the interactive setting since the data sent by Alice may depend on some data sent by Bob in a previous flow, which makes both the order and number of flows important in analysis.

Let $\mathcal{M}$ be the space of messages. In a Message Authentication Protocol, the claimant Alice chooses a message $M \in \mathcal{M}$ and sends it to Bob using the protocol. At the end, Bob either outputs (Alice, M'), where $M' \in \mathcal{M}$, or he rejects.

Consider a randomized algorithm *split* : $\mathcal{M} \to \mathcal{M}_1 \times \mathcal{M}_2$ which takes any message M as input and maps it into a pair (m_1, m_2), where m_1 is shorter than m_2. The reverse procedure is carried out by a deterministic algorithm *reconstruct* : $\mathcal{M}_1 \times \mathcal{M}_2 \to \mathcal{M} \cup \{\perp\}$ which takes a pair (m_1, m_2) and maps it into a message $M \in \mathcal{M}$ or a "reject" sign $\perp$.

In order to employ the *split* and *reconstruct* algorithms in a Message Authentication Protocol, we need them to satisfy the following requirements:

(i) Correctness property: Any message can be uniquely recovered. That is, for any $M \in \mathcal{M}$,
$$\mathit{reconstruct}(\mathit{split}(M)) = M.$$

(ii) Binding property: The Binding game of Fig. 2.1 is hard. In other words, it is computationally infeasible to find a message M such that given (m_1, m_2), where $\mathit{split}(M) = (m_1, m_2)$, one can efficiently find an $m_2' \in \mathcal{M}_2 \setminus \{m_2\}$ so that
$$\mathit{reconstruct}(m_1, m_2') \neq M \quad \text{and} \quad \mathit{reconstruct}(m_1, m_2') \in \mathcal{M}$$
with non-negligible probability.

$$\boxed{\begin{array}{lcl}
\underline{\text{Oscar}} & & \underline{\text{challenger}} \\
\text{Choose } M \in \mathcal{M} & \xrightarrow{\quad M \quad} & \text{Compute } \mathit{split}(M) = (m_1, m_2) \\
 & \xleftarrow{\quad m_1, m_2 \quad} & \\
\text{Send } m_2'. & \xrightarrow{\quad m_2' \quad} & \text{Compute } M' = \mathit{reconstruct}(m_1, m_2'). \\
 & & \text{Oscar wins if } M' \in \mathcal{M} \text{ and } M \neq M'
\end{array}}$$

Figure 2.1: The Binding Game

Given a pair (m_1, m_2) corresponding to a message M, it is desirable that for all m_2' either $\mathit{reconstruct}(m_1, m_2') = M$ or $\mathit{reconstruct}(m_1, m_2') = \perp$ with high probability. The Binding property ensures that the values m_1 and m_2 are bound in such a way that for almost all values of m_2', the pair (m_1, m_2') corresponds to the same message M or it is going to be rejected.

We define a pair of algorithms $(\mathit{split}, \mathit{reconstruct})$ to be (T, ϵ)-binding if any adversary bounded by a time complexity T wins the Binding game with a probability of success at most ϵ.

Now consider the following general non-interactive Message Authentication Protocol, where the *split* and *reconstruct* algorithms satisfy the correctness property and are (T, ϵ)-binding. This protocol, abbreviated as GNIMAP, is also depicted in Fig. 2.2.

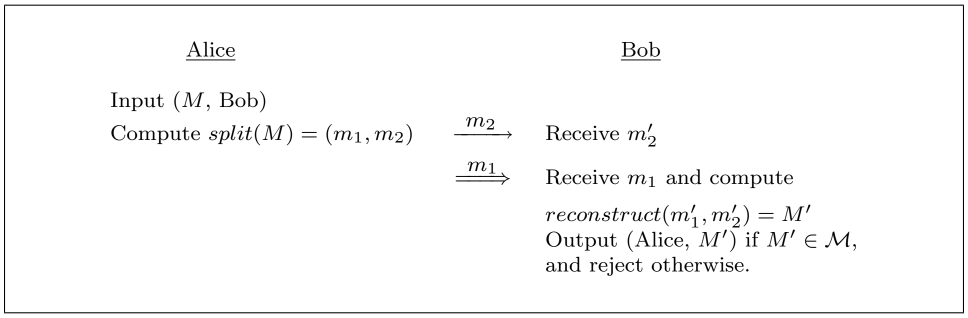

Alice
Bob
Input (M, Bob)
Compute $\mathit{split}(M) = (m_1, m_2)$
$\xrightarrow{m_2}$
Receive m_2'
$\xRightarrow{m_1}$
Receive m_1 and compute
$\mathit{reconstruct}(m_1', m_2') = M'$
Output (Alice, M') if $M' \in \mathcal{M}$,
and reject otherwise.

Figure 2.2: General Non-Interactive Message Authentication Protocol

General Non-Interactive Message Authentication Protocol (GNIMAP):

1. On input (M, Bob), Alice computes $\mathit{split}(M) = (m_1, m_2)$.
2. Alice sends m_2 to Bob over the broadband channel.
3. Bob receives m_2'.[1]
4. Alice sends m_1 to Bob over the authenticated channel.
5. Bob receives m_1 from Alice.
6. Bob computes $\mathit{reconstruct}(m_1', m_2') = M'$.
7. Bob outputs (Alice, M') if $M' \in \mathcal{M}$, and rejects otherwise.

[1]Note that the values that Bob receives on the broadband channel might have been altered by an adversary. Hence, we use the notation D' in the receiving end where the data D is transmitted.

2.1.1 Attack Model

The correctness of the aforementioned GNIMAP is ensured by property (i). In other words, Bob can successfully recover M from the protocol if all the participants have been honest and no attack has occurred. In order to analyze the security of GNIMAP, we need to define an attack model. The adversarial goal and capabilities are described in the following section.

In the setting of message authentication protocols, the *adversarial goal* is to make Bob accept a message M along with the identity of Alice, when he was supposed to reject (that is, when the message M was never sent by Alice to Bob). There are two main types of attacks to consider: *impersonation* attacks and *substitution* attacks.

In an impersonation attack, the attacker tries to convince Bob that a message M is sent from Alice, while in fact M was never sent from Alice and the session has been initiated by the adversary. Figure 2.3 depicts the impersonation attack in the setting of GNIMAP.

Note that according to our model, the adversary cannot modify the data sent over the authenticated channel, but he or she can replay them. Hence, the authenticated flow in an impersonation attack is a replay of a previous flow sent by Alice.

Eve		Bob
Choose m_2'	$\xrightarrow{m_2'}$	
Let $m_1' = m_1$, where Alice has sent m_1 in a previous flow	$\xRightarrow{m_1'}$	Compute $M' = \mathit{reconstruct}(m_1', m_2')$. If $M' \in \mathcal{M}$, then output (Alice, M'), reject otherwise.

Figure 2.3: An Impersonation Attack Against GNIMAP

In a substitution attack, on the other hand, Alice initiates a session with Bob trying to send him a message M. The adversary then substitutes M' instead of M. So, Bob receives M' and not M. The adversary may have changed part or all of M to get M'. In case of our protocol, the adversary replaces m_2 with m_2', after Alice splits M into (m_1, m_2). The authenticated value m_1 cannot be substituted according to the model.

Note that the message M might have been chosen by the adversary. In other words, the adversary can make Alice send a message that the adversary has chosen.

This ability of the adversary may not be considered in all models. We do consider it in our model since it makes the adversary stronger and results in a stronger model. Figure 2.4 illustrates a substitution attack against GNIMAP.

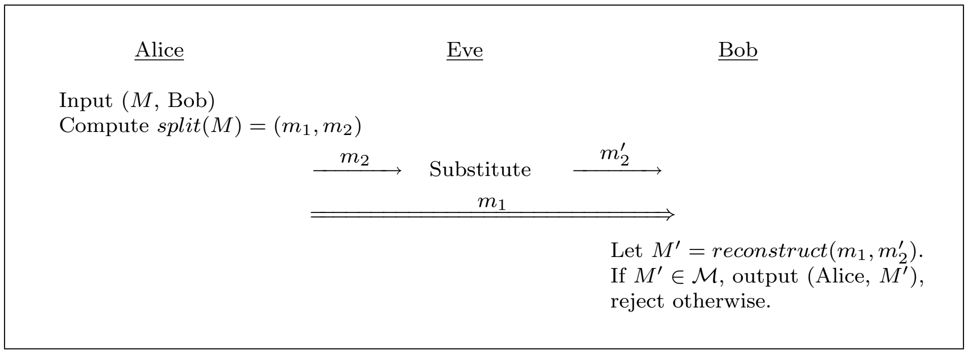

Alice
Eve
Bob
Input (M, Bob)
Compute $split(M) = (m_1, m_2)$
m_2
Substitute
m_2'
m_1
Let $M' = reconstruct(m_1, m_2')$.
If $M' \in \mathcal{M}$, output (Alice, M'),
reject otherwise.

Figure 2.4: A Substitution Attack Against GNIMAP

One could argue that since an attacker has to use a previous flow in an impersonation attack, the attack should not be called an "impersonation", and should be called a substitution; see for instance [GN04]. However, we believe that allowing the adversary to replay previous authenticated flows in an impersonation attack results in a stronger adversary and, ultimately, a stronger model. Moreover, despite the fact that the two attack scenarios are equivalent in the non-interactive setting, they result in two very different attack scenarios in the interactive setting, see for instance [MS08a].

We consider an adaptive chosen plain-text attack (ACPA) model in our general setting. Note that the ACPA model is very strong and desirable compared to other models. An adaptive chosen plaintext attack consists of two stages: an *information gathering* stage and a *deception* stage.

The model presumes that in the information gathering stage, the attacker has the capability to adaptively choose a number of arbitrary messages M_i, and have Alice send them to Bob. The attacker then records the communication for further use. He or she can choose the subsequent messages to be sent by Alice using the results of the messages already sent. Note that the description of the *split* function is known to the adversary. Hence, the adversarial goal is to gather as many authenticated flows as possible in this stage. In addition, we assume that the attacker has precomputing capabilities and is able to mount "dictionary"-type attacks. The information gathering stage of an attack against GNIMAP is depicted in Fig. 2.5.

Let $\mathcal{N}$ denote the set of all messages M sent by Alice to Bob before the start of deception stage, and the set N denote the set of ordered pairs (m_1, m_2) sent by

Alice to Bob over the two channels before the start of deception stage. Note that the set $\mathcal{N}$ includes all messages previously sent by Alice to Bob with or without the request of the attacker.

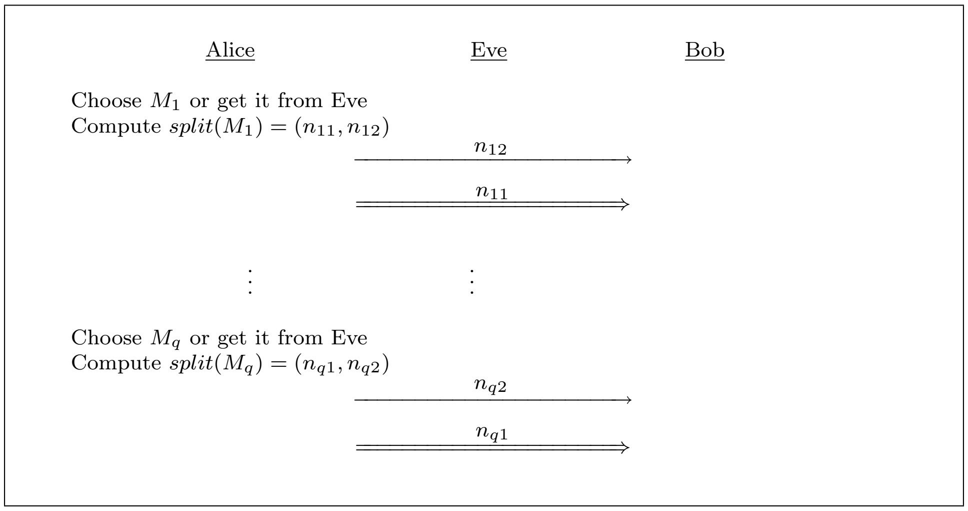

Alice
Eve
Bob
Choose M_1 or get it from Eve
Compute $split(M_1) = (n_{11}, n_{12})$
n_{12}
n_{11}
Choose M_q or get it from Eve
Compute $split(M_q) = (n_{q1}, n_{q2})$
n_{q2}
n_{q1}

Figure 2.5: Information Gathering Phase of an Attack

We use the term *querying complexity* of an adversary to refer to the number q of messages sent by Alice to Bob during the information gathering stage. On the other hand, the term *offline complexity* is used to refer to the computational complexity T of an adversary.

The deception stage is where the attack occurs. That is, the adversary tries to achieve his or her goal by making Bob accept a message M along with the identity of Alice, when he was supposed to reject. The attack is either a substitution or an impersonation attack.

In case of a substitution attack, Alice is sending a pair (m_1, m_2) to Bob. The adversary substitutes m_2 with m_2' and leaves m_1 untouched. Now let M be one of the messages sent by Alice in the information gathering stage. On the other hand, consider an impersonation attack where the adversary sends m_2' and replays m_1. Given that $M \in \mathcal{N}$, this impersonation attack is equivalent to the substitution attack that we started with. This fact is illustrated in Fig. 2.6. Hence, without loss of generality, we only consider impersonation attacks in the deception phase.

In the deception stage, the attacker tries to impersonate Alice by sending a single message $M' \notin \mathcal{N}$. The attack succeeds if Bob accepts, and it fails otherwise. In choosing M' the attacker can use all the information obtained from the information gathering stage, which includes the messages sent previously by Alice without the attacker's request. The deception stage is illustrated in Fig. 2.7.

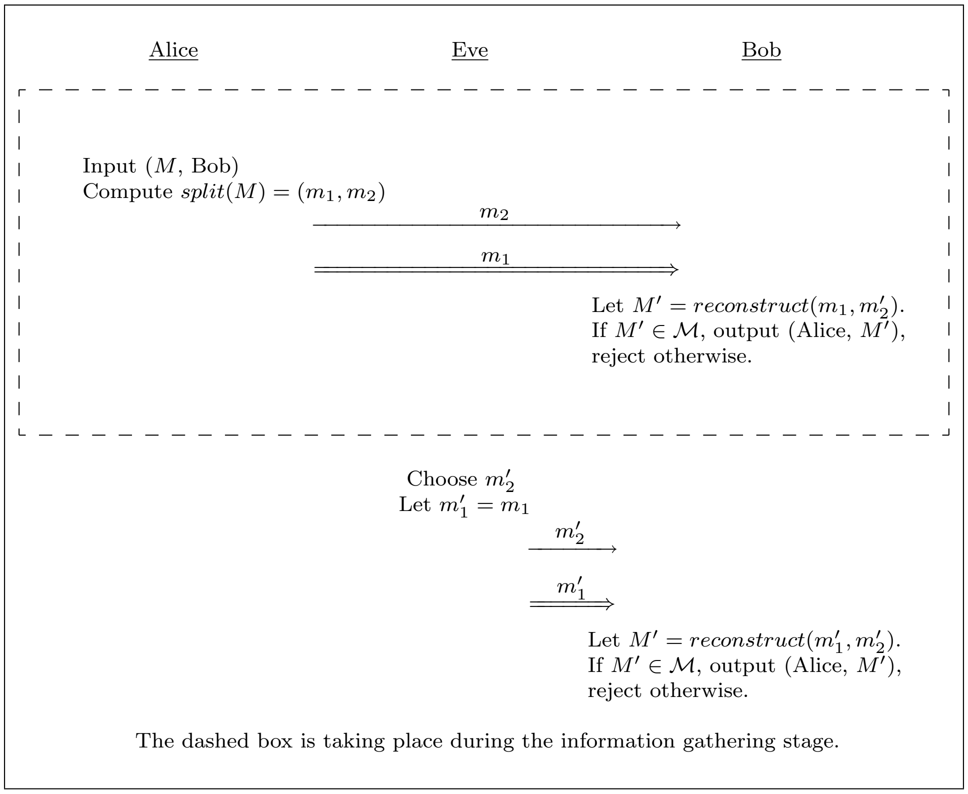

Alice
Eve
Bob
Input (M, Bob)
Compute $split(M) = (m_1, m_2)$
m_2
m_1
Let $M' = reconstruct(m_1, m_2')$.
If $M' \in \mathcal{M}$, output (Alice, M'),
reject otherwise.
Choose m_2'
Let $m_1' = m_1$
m_2'
m_1'
Let $M' = reconstruct(m_1', m_2')$.
If $M' \in \mathcal{M}$, output (Alice, M'),
reject otherwise.
The dashed box is taking place during the information gathering stage.

Figure 2.6: Equivalence of Impersonation and Substitution Attacks in GNIMAP.

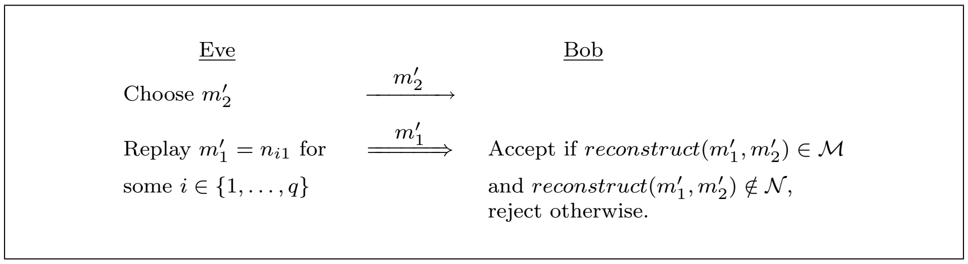

Eve
Bob
Choose m_2'
m_2'
Replay $m_1' = n_{i1}$ for
some $i \in \{1, \ldots, q\}$
m_1'
Accept if $reconstruct(m_1', m_2') \in \mathcal{M}$
and $reconstruct(m_1', m_2') \notin \mathcal{N}$,
reject otherwise.

Figure 2.7: Deception Phase of an Attack

Note that anyone can replay both flows of a previous conversation between Alice and Bob. In this case, Bob accepts a message that was previously sent by Alice. However, this replay impersonation does not constitute an attack. In a successful attack, the adversary is required to replay the authenticated flow and change the information sent over the insecure channel. The first flow could be a replay of a previously transmitted first flow. However, the two flows of the attack should not be identical to a previous conversation of Alice and Bob, otherwise the “attack” is considered a replay.

2.1.2 Security Analysis

In this section, we prove that GNIMAP is secure given the properties enumerated in Section 2.1 and under the attack model described in Section 2.1.1. The proof is based on a reduction.

Associated to each attack, there are sets $\mathcal{N}$ and N, resulting from the information gathering stage, and a pair (m'_1, m'_2), from the deception stage, according to our attack model. Let $\mathcal{N} = \{M_1, M_2, \ldots, M_q\}$. Then, for each $1 \leq i \leq q$

$$N = \{(n_{i1}, n_{i2}) : 1 \leq i \leq q\} \subset \mathcal{M}_1 \times \mathcal{M}_2, \text{ where } \mathit{reconstruct}(n_{i1}, n_{i2}) = M_i.$$

The pair (m'_1, m'_2) corresponds to the deception stage, where the adversary replays m'_1 over the authenticated channel, and sends m'_2 over the insecure channel.

Let us assume that an attack has occurred and Bob has accepted. That is, the adversary has impersonated Alice by sending the pair (m'_1, m'_2) to Bob. Moreover, Bob has accepted and has output $(M'$, Alice), where $M' = \mathit{reconstruct}(m'_1, m'_2)$.

In any successful attack, the adversary needs to replay the authenticated flow. As a result, $m'_1 \in \{n_{11}, n_{21}, \ldots, n_{q1}\}$. That is $m'_1 = n_{i1}$, for some $1 \leq i \leq q$. Let i be the smallest index for which $m'_1 = n_{i1}$. Moreover, $M' \notin \{M_1, M_2, \ldots, M_q\}$, since otherwise the attack is only a replay and not a real attack.

We now formally prove that the GNIMAP is secure given that $(\mathit{split}, \mathit{reconstruct})$ is (T, ϵ)-binding. That is, we reduce an adversary who can attack the GNIMAP with non-negligible probability to an adversary who wins the Binding game with non-negligible probability.

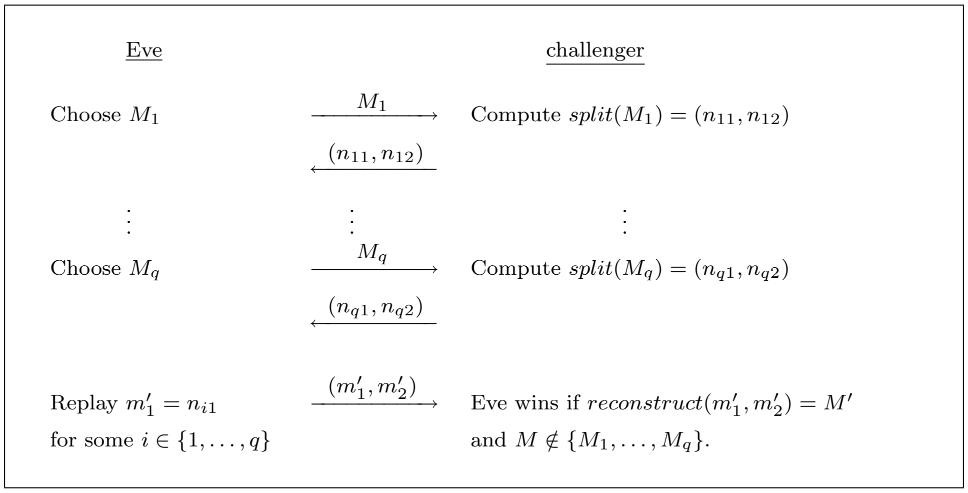

Eve
challenger
Choose M_1
M_1
Compute $\mathit{split}(M_1) = (n_{11}, n_{12})$
(n_{11}, n_{12})
Choose M_q
M_q
Compute $\mathit{split}(M_q) = (n_{q1}, n_{q2})$
(n_{q1}, n_{q2})
Replay $m'_1 = n_{i1}$ for some $i \in \{1, \ldots, q\}$
(m'_1, m'_2)
Eve wins if $\mathit{reconstruct}(m'_1, m'_2) = M'$ and $M \notin \{M_1, \ldots, M_q\}$.

Figure 2.8: GNIMAP Game

Consider the game depicted in Fig. 2.8. We call this game the "GNIMAP game". This is because, if Eve wins this game with probability ϵ, then the game translates into an attack against GNIMAP with success probability ϵ. Here, Eve is facing a challenger who is simulating both Alice and Bob. The game consists of q rounds of Eve sending messages M_i and the challenger responding with (n_{i1}, n_{i2}), where $split(M_i) = (n_{i1}, n_{i2})$. These q rounds correspond to the information gathering phase of the attack. The last round is analogous to the deception phase where Eve sends her pair (m'_1, m'_2). Eve wins the game if $m'_1 = n_{i1}$, for some $i \in \{1, \ldots, q\}$, while $reconstruct(m'_1, m'_2) = M' \neq M_i$.

Assuming that Eve wins this game with non-negligible probability, we can employ her in the Binding game of Fig. 2.1.

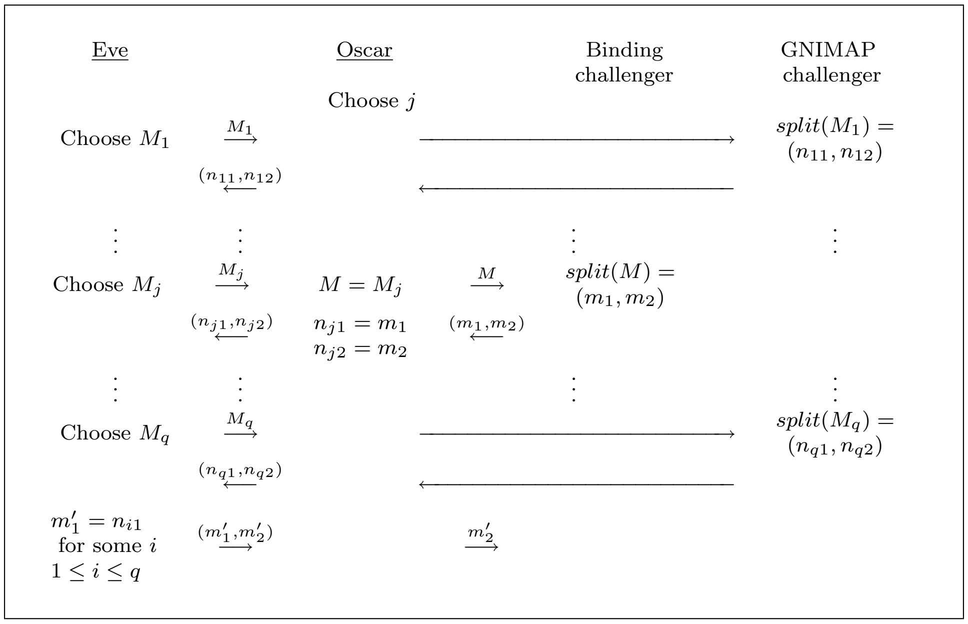


Figure 2.9: Reducing the GNIMAP Game to the Binding Game

Depicted in Fig. 2.9, Eve is playing against her GNIMAP game challenger (which is simulated by Oscar), while Oscar is playing against his Binding game challenger. Oscar will use the results of the GNIMAP game to win his Binding game. He first chooses a random value $j \in_R \{1, \ldots, q\}$. Then, Eve will carry out her own attack against the GNIMAP challenger. That is, Eve sends messages M_t and receives n_{t1} and n_{t2}.

The responses, n_{t1} and n_{t2}, come from computing $split(M_t)$, except when $t = j$. In the jth round, Oscar forwards $M = M_j$ to his challenger. The challenger responds with a pair (m_1, m_2). Then, Oscar forwards $n_{j1} = m_1$ and $n_{j2} = m_2$ to Eve.

After q rounds, Eve chooses a message M' and sends m'_1 and m'_2. Note that for Eve to win, $m'_1 = n_{i1}$ for some $i \in \{1, \ldots, q\}$. Oscar simply forwards m'_2 to his challenger if $j = i$, and quits otherwise.

Note that from Eve's point of view, this game is no different than the game of Fig. 2.8.

Assuming that Eve wins her game with probability p, Oscar clearly wins his game with probability p/q. Hence, we have proved the following Theorem.

Theorem 1. *Assume that there is a GNIMAP where the pair $(split, reconstruct)$ is (T, ϵ)-binding. In the ACPA model, any adversary against this GNIMAP with querying complexity q and offline complexity T has a probability of success p at most $q\epsilon$.*

We note that our reduction is not tight. However, it is normal to assume that $q \leq 2^{10}$ in manual authentication scenarios.[2]

2.2 Previous NIMAPs

In this section, we first define the kind of hash functions that are going to come up in our discussion. Secondly, we briefly introduce the previous NIMAPs found in the literature. Then, the security of these protocols is analyzed with respect to our general model.

We use the following definitions of different types of hash functions in the rest of the chapter.

A **Collision Resistant (CR) Hash Function** H, is a hash function where it is hard to find distinct elements x and y such that $H(x) = H(y)$. The pair (x, y) is called a collision pair. For security purposes, the length of the hash value is required to be more than $2k$ bits, where output size of H is k. Otherwise, an adversary has a good chance of finding a collision pair using an offline birthday attack.

A **Second-Preimage Resistant (SPR) Hash Function** H, is a hash function where given a value x, it is hard to find a value y, $x \neq y$, such that $H(x) = H(y)$. In this case, the best generic attack is exhaustive search. Hence, the length of the hash value is required to be at least k bits.

[2]The reduction proposed by Pasini and Vaudenay [PV06] is also not tight and they get the same probability of success, p/q. They also assume that $q \leq 2^{10}$.

An ϵ-**Universal Hash Function Family, (ϵ-UHFF)** H is a collection of functions H_K depending on a random key K, where $\Pr[H_K(x) = H_K(y)] \leq \epsilon$ for any two distinct values x and y, where the probability is taken over the choices of K.

We now briefly summarize three NIMAPs found in the literature.

Balfanz-Smetters-Stewart-Wong NIMAP Balfanz et al. [BSSW02] introduced the idea of hashing the data to be authenticated and delivering the hash value in an authenticated way to the verifier. Their protocol is based on a collision resistant hash function. It is depicted in Fig. 2.10.

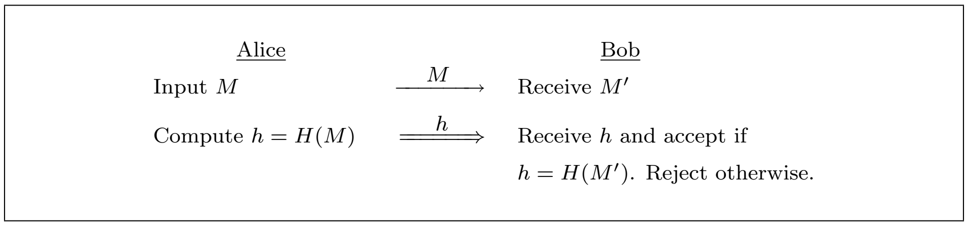


Figure 2.10: Balfanz et al. NIMAP

The adversary can work offline and find a collision M_1 and M_2 yielding the same hash value. Then, M_1 is given to Alice in the information gathering stage and she sends Bob the value of $H(M_1)$ over the authenticated channel. The adversary replays this authenticated flow along with M_2 and makes Bob accept. This attack is depicted in Fig. 2.11. If the adversary can mount the above attack efficiently, then this protocol fails to satisfy property (ii) of Section 2.1.

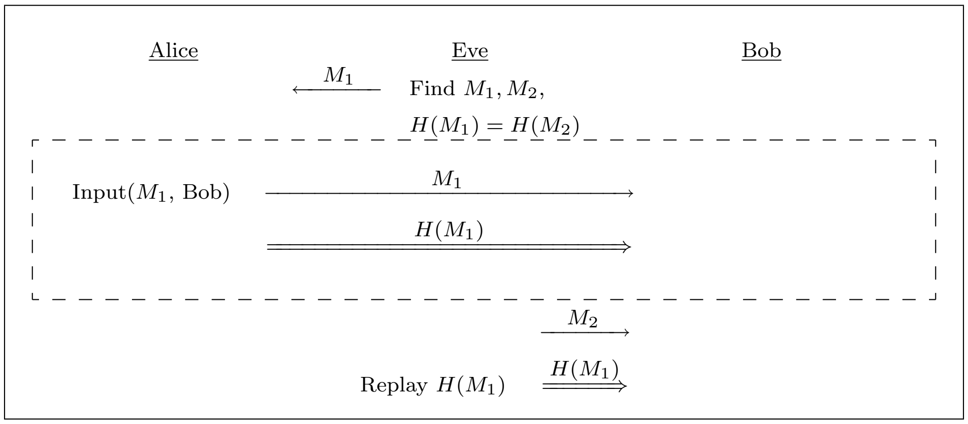


Figure 2.11: Attack against the Balfanz et al. NIMAP

The collision pair, M_1 and M_2, could be found using a "birthday attack". Birthday attacks have square root complexity. If we consider algorithms of complexity

2^{80} inefficient, then in order to make this attack not efficient we need to increase the size of the authenticated bits, that is, h, to 160 bits.

Gehrmann-Mitchell-Nyberg MAP: MANA I Gehrmann et al. [GMN04] introduced MANA I, for *manual authentication*, based on an ϵ-universal hash function family H. The original form of this protocol is not a NIMAP. Later, Vaudenay [Vau05] proposed a non-interactive version of MANA I. This protocol is depicted in Fig. 2.12. In the original proposal, confidentiality of the authenticated channel is required. This requirement is very restrictive in general. Vaudenay [Vau05] has proved that a "stall-free" authenticated channel is enough to ensure the security of MANA I. In a stall-free channel, once a message is sent, it is either received by the recipient right away or it is never received. In other words, it is not possible to delay a flow under this assumption. However, the stall-free requirement is still quite strong and may not be realistic in an arbitrary authenticated channel.

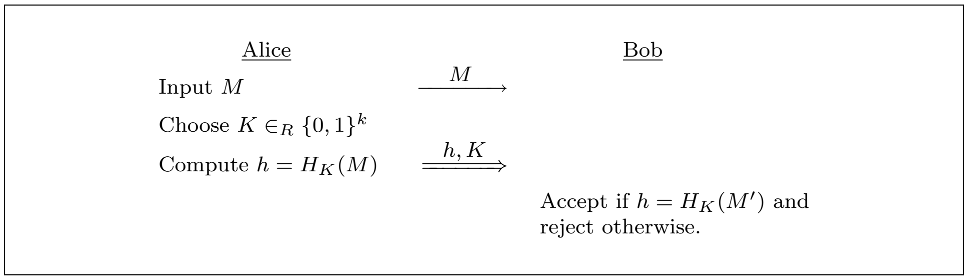


Figure 2.12: MANA I

According to our model, the adversary can record a pair $(H_K(M), K)$ from the information gathering stage and find M' such that $H_K(M) = H_K(M')$. The adversary then sends M' over the insecure channel and replays $(H_K(M), K)$ over the authenticated channel. (Note that this attack will not work in a stall-free channel.) Having recorded K, finding M' such that $H_K(M) = H_K(M')$ is usually an easy computation. This is because the function H_K is a member of a universal hash family and typically it has a simple structure[3].

Pasini-Vaudenay NIMAP Pasini and Vaudenay [PV06] proposed a NIMAP, illustrated in Fig. 2.13, based on Second-Preimage Resistant hash functions. The protocol is in the Common Reference String (CRS) model, which assumes a random string K_p has been previously distributed to everyone. The *commit* function has

[3]For example, here is one commonly used universal hash family. Let p be a large prime and let $n < p$. For all pairs $K = (a, b) \in \mathbb{Z}_p{}^* \times \mathbb{Z}_p$, define a hash function $h_K(x) = (ax + b \bmod p) \bmod n$. The family $\{h_K\}$ is a universal hash family.

two inputs: the message M and the CRS K_p. It outputs a commit value c and a decommit value d. This function is non-deterministic and is playing the role of the *split* algorithm. The *open* algorithm, on the other hand, is a deterministic algorithm. It uniquely outputs M on input (K_p, c, d).

$$\begin{array}{lcl}
\underline{\text{Alice}} & & \underline{\text{Bob}} \\
\text{input } M & & \\
(c, d) \leftarrow commit(K_p, M) & \xrightarrow{(c\|d)} & M' \leftarrow open(K_p, c', d') \\
\text{Compute } h = H(c) & \xRightarrow{h} & \text{Accept if } h = H(c') \text{ and} \\
 & & \text{reject otherwise.}
\end{array}$$

Figure 2.13: Pasini-Vaudenay NIMAP

An adversary attacking the NIMAP is reduced to an adversary who finds second-preimages or breaks the trapdoor of the commitments, details can be found in [PV06]. To achieve security against an adversary with complexity 2^{80} and $q = 2^{10}$, they need to authenticate 100 bits. More details can be found in [PV06].

There is always the issue of authenticity attached to public parameters such as K_p. Hence, it possibly restricts the application of this NIMAP. Moreover, we are trying to replace the use of any PKI by using NIMAPs. As a result, this protocol does not seem to be the optimal solution.

On the other hand, this NIMAP is based on the assumption that trapdoor commitment schemes exist, as well as SPR hash functions. This protocol satisfies the properties of Section 2.1.

2.3 A New Computationally Secure NIMAP

In this section, we first define Hybrid-Collision Resistance for hash functions. Secondly, we discuss the difficulty of finding hybrid-collisions. Moreover, a new NIMAP based on Hybrid-Collision Resistant hash functions is introduced. The security of this NIMAP is ensured by showing that it satisfies the properties we listed in Section 2.1 when using Hybrid-Collision Resistant hash functions.

2.3.1 Hybrid-Collision Resistant Hash Function

We define a **Hybrid-Collision Resistant (HCR) Hash Function** H, to be a hash function in which the game of Fig. 2.14 is hard, for fixed values l_1 and l_2.

Moreover, we say H is a (T, ϵ)-HCRHF if an adversary with complexity T wins the game of Fig. 2.14 with probability at most ϵ.

Oscar		challenger
Choose M, $\lvert M\rvert = l_1$.	$\xrightarrow{M}$	Choose $K \in_R \{0,1\}^{l_2}$.
	$\xleftarrow{K}$	
Choose L, $\lvert L\rvert = l_1 + l_2$.	$\xrightarrow{L}$	Oscar wins if $L \neq M\|K$ and $H(M\|K) = H(L)$.

Figure 2.14: HCR Game

Furthermore, we call the pair $(L, M\|K)$ a *hybrid-collision.* Note that if $l_2 = 0$, then HCR is equivalent to CR. On the other hand, HCR is very close to SPR when $l_1 = 0$. (The lengths are fixed in the HCR game.) In fact, HCR is interpolating between CR and SPR. This suggests that finding hybrid-collisions is at least as hard as collisions, but at most as hard as second-preimages. We will investigate this matter in more detail in the next section.

2.3.2 On the Difficulty of the HCR Game

As far as we know, the problem of finding hybrid-collisions has not been previously addressed in the literature. Here, we investigate this problem in the random oracle model. This gives us an intuition about the difficulty of the problem compared to finding collisions or second-preimages.

Let H be a hash function randomly chosen from $\mathcal{F}^{\mathcal{X},\mathcal{Y}}$ (the set of all functions from $\mathcal{X}$ to $\mathcal{Y}$), where $\mathcal{X} = \{0,1\}^{l_1+l_2}$ is the set of all possible binary strings of length $l_1 + l_2$ and $|\mathcal{Y}| = 2^k$. Assume that we are only permitted oracle access to H, i.e., the only way to compute $H(x)$ is to query the value x to the oracle. Further, assume that the adversary, Oscar, is able to access the random oracle T times, where $T = 2^t$.

In order to analyze the difficulty of the HCR game, we find an upper bound on the probability ϵ of Oscar winning the HCR game.

Let distinct random values $X_1, X_2, \ldots, X_T$ be Oscar's inputs to the random oracle. Moreover, let the hybrid-collision be $(L, M\|K)$. We write $X_i = M_i\|K_i$, where $|K_i| = l_2$ and $|M_i| = l_1$, for all $i = 1, \ldots, T$.

Let D denote the event that $M\|K$ is equal to one of $X_1, ..., X_T$, and let E denote the event that $M\|K$ collides with some X_i (i.e. $H(M\|K) = H(X_i)$ where

$M \| K \neq X_i$). We are interested in computing an upper bound on $\Pr[E]$. We will do this by conditioning on the event D:

$$\begin{aligned}\Pr[E] &= \Pr[\neg D] \times \Pr[E|\neg D] + \Pr[D] \times \Pr[E|D]\\ &\leq \Pr[E|\neg D] + \Pr[D] \times \Pr[E|D]\\ &= \Pr[E|\neg D] + \Pr[D \text{ and } E].\end{aligned}$$

Denote $\epsilon_1 = \Pr[E|\neg D]$ and $\epsilon_2 = \Pr[D \text{ and } E]$. We will compute upper bounds on ϵ_1 and ϵ_2.

Given $\neg D$, the probability that $H(M\|K) = H(X_j)$ for any given j is 2^{-k}. Hence, the probability of occurrence of at least one collision is $\epsilon_1 = 1-(1-2^{-k})^T$. If $T = 2^t$ is small compared to 2^k, then ϵ_1 is approximately 2^{t-k}.

We bound ϵ_2 as follows. Construct a graph G with $V(G)$ and $E(G)$, denoting the set of vertices and edges respectively, where $V(G) = \{X_1, X_2, \ldots, X_T\}$. Moreover, for any m and n, $m \neq n$, $X_m X_n \in E(G)$ if and only if $H(X_m) = H(X_n)$. Now define $V' = \{X_i \in V(G) : \deg(X_i) \geq 1\}$. It is clear that $\epsilon_2 = \Pr[M\|K \in V']$.

Let $\text{Exp}[|V'|]$ denote the expectation of $|V'|$. Now, since K is a random bitstring of length l_2, $\Pr[M\|K \in V'] \leq \text{Exp}[|V'|] \times 2^{-l_2}$, where the expectation is taken over the choices of K and H.

Note that the maximum number of edges of G is $T^2/2$. Furthermore, for any randomly chosen X_m and X_n, the probability that $X_m X_n$ is an edge is 2^{-k}. Hence, the expected number of edges of G is $2^{-k}T^2/2 = 2^{2t-k-1}$. In addition, the expected number of vertices of positive degree is at most 2^{2t-k}. As a result, $\text{Exp}[|V'|] \leq 2^{2t-k}$. Therefore, $\epsilon_2 \leq 2^{2t-k-l_2}$.

Let C denote the event that Oscar wins the HCR game. Now, we compute

$$\begin{aligned}\Pr[\mathrm{C}] &= \Pr[\neg E] \times \Pr[\mathrm{C}|\neg E] + \Pr[E] \times \Pr[\mathrm{C}|E]\\ &\leq \Pr[\mathrm{C}|\neg E] + \Pr[E]\\ &= 2^{-k} + \epsilon_1 + \epsilon_2\\ &\leq (2^t+1)2^{-k} + 2^{2t-k-l_2}\\ &\approx 2^{t-k} + 2^{2t-k-l_2}.\end{aligned}$$

Note that the length of the original message, l_1, has no influence in the analysis in the random oracle model. However, once a concrete hash function is chosen, the amount of time it takes to compute a hash function is in proportion to the size of the input, and as a result, the size of the message will be a factor to consider. The shorter the messages are, the more hash function computations can be handled in a fixed amount of time.

In Section 2.3.4 we examine p, the overall success probability of the adversary, given particular values for parameters k, t and l_2.

2.3.3 A new NIMAP based on HCR hash functions.

Let H be an HCR hash function and fix k, l_1, and l_2. Now, consider the following proposed NIMAP.

1. On input (M, Bob), $|M| = l_1$, Alice chooses $K \in_R \{0,1\}^{l_2}$ uniformly at random.
2. Alice sends (M, K) to Bob over the broadband channel.
3. Bob receives (M', K'), where $|M'| = l_1$ and $|K'| = l_2$.
4. Alice computes $h = H(M\|K)$ and sends h to Bob over the authenticated channel.
5. Bob receives h from Alice.
6. Bob outputs (Alice, M') if $h = H(M'\|K')$, and rejects otherwise.

The above NIMAP is also depicted in Fig. 2.15.

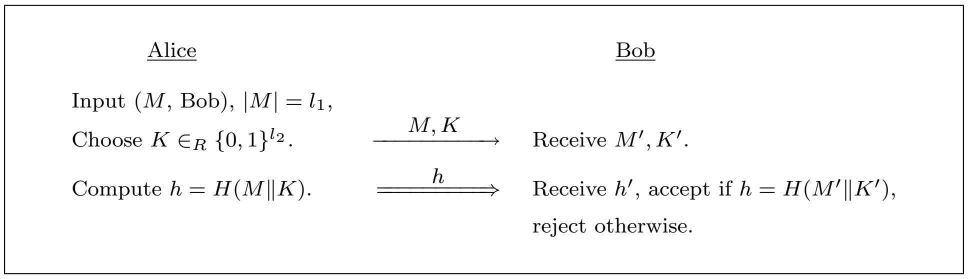

Alice
Bob
Input (M, Bob), $|M| = l_1$,
Choose $K \in_R \{0,1\}^{l_2}$.
$\xrightarrow{M, K}$
Receive M', K'.
Compute $h = H(M\|K)$.
$\xRightarrow{h}$
Receive h', accept if $h = H(M'\|K')$,
reject otherwise.

Figure 2.15: The New NIMAP

In this NIMAP, $m_1 = H(M\|K) = h$ and $m_2 = (M, K)$ for a random key K. Moreover, for any M', K' and h, *reconstruct*$(h, (M', K')) = M'$ if $h = H(M'\|K')$, and *reconstruct*$(h, (M', K')) = \perp$ otherwise.

Clearly, this $(\mathit{split}, \mathit{reconstruct})$ satisfies the Property (i) of Section 2.1. That is, any message M can be uniquely recovered:

$$\mathit{reconstruct}(\mathit{split}(M)) = \mathit{reconstruct}(H(M\|K), (M, K)) = M.$$

Next we need to show that our $(\mathit{split}, \mathit{reconstruct})$ satisfies the Property (ii) of Section 2.1 which says: It is computationally infeasible to find a message M such that given (m_1, m_2), where $\mathit{split}(M) = (m_1, m_2)$, one can efficiently find an $m_2' \in \mathcal{M}_2 \setminus \{m_2\}$ so that $\mathit{reconstruct}(m_1, m_2') \in \mathcal{M}$ with non-negligible probability.

We substitute for the *split* and *reconstruct* algorithms and restate the Binding Property for our NIMAP as follows:

It is computationally infeasible to find a message M, $|M| = l_1$, such that given $H(M\|K)$ and K, $K \in_R \{0,1\}^{l_2}$, one can efficiently find an L of size $l_1 + l_2$, $L \neq M\|K$, so that $H(L) = H(M\|K)$ with non-negligible probability.

This is implied from the assumption that the HCR game is hard. Note that the binding property for our NIMAP translates to HCR game being hard, but the opposite is not true and does not need to hold for our application. In other words, Oscar may win the HCR game by finding a collision of the form $(M\|K', M\|K)$, with $K \neq K'$. However, this collision does not constitute an attack against our NIMAP since the messages are the same. On the other hand, all instances of a successful attack against our NIMAP translate into a winning strategy against the HCR game.

Assuming that H is a (T, ϵ)-HCRHF, we conclude that $(\mathit{split}, \mathit{reconstruct})$ of this NIMAP is (T, ϵ)-binding. Hence, we get the following Corollary of Theorem 1.

Corollary 1. *Let H be a (T, ϵ)-HCRHF. Any adversary against the NIMAP of Fig. 2.15, with querying complexity q and offline complexity T, has a probability of success p at most $q\epsilon$.*

Note that we do not require any public parameters to be distributed ahead of time. One could argue that the description of the HCR hash function needs to be distributed in an authentic manner ahead of time. In practice, however, these protocols are going to use standard built-in hash functions which do not require any authentication of public parameters, which would be required for commitment schemes.

Our new protocol looks similar to the protocol of Fig. 2.12 with the difference that K is moved from the authenticated channel to the broadband channel.

However, there are several differences. The underlying hash function security requirement is different, properties of the channels are different, and the resulting overall security of our protocol is different from those of MANA I and its NIMAP version depicted in Fig. 2.12. We do not assume that our channels provide confidentiality or are stall-free. The notion of hybrid-collision resistance is different from the security notion of ϵ-universal hash function families.

In our protocol, ℓ_1, which is the length of the messages being authenticated, is a fixed parameter. That is, our protocol only authenticates messages of fixed length. However, note that M is being sent over the broadband channel and sending long messages over this channel is very cheap. As a result, we can set ℓ_1 large enough for the desired application, and pad short messages with a one followed by enough zeros, if necessary.

2.3.4 Parameter sizes

Let $T = 2^t$ and q be the offline and querying complexities respectively. That is, the adversary is allowed to use T hash computations and make Alice send q messages to Bob. Moreover, let H be a (T, ϵ)-HCRHF and let k be the size of H.

According to Corollary 1, an adversary attacking our proposed NIMAP, using T hash computations and q messages, has probability of success $p \leq q\epsilon$.

Pasini and Vaudenay [PV06] assume that $q \leq 2^{10}$ and $t \leq 70$. They also require the probability of success of the adversary against the protocol of Fig. 2.13 be less that 2^{-20}. For this to happen, one needs to authenticate 100 bits. That is $k = 100$.

Using the same parameters, $q \leq 2^{10}$, $t \leq 70$, and $k = 100$ we obtain that $\epsilon \approx 2^{-30} + 2^{40-l_2}$. In order to achieve the same level of security obtained by Pasini and Vaudenay [PV06], i.e., $p \leq 2^{-20}$, we should have $\epsilon \approx 2^{-30}$. Thus, if we let l_2 large enough, e.g., $l_2 \geq 80$, in our protocol of Fig. 2.15, then we obtain the same level of security of the protocol of Fig. 2.13. That is, the amount of information sent over the authenticated channel is the same as in the Pasini-Vaudenay protocol.

2.3.5 Advantages of the proposed NIMAP

Our proposed NIMAP of Fig. 2.15 benefits from a simple and easy to implement structure. It is based on a single assumption that HCR hash functions exist. Note that any CR hash function satisfies the HCR notion.

We do not use any commitment scheme or require any public parameters available to users such as the CRS. The amount of information sent over the authenticated channel is as low as the most secure NIMAP proposed so far, while achieving the same level of security.

2.4 On Unconditionally Secure NIMAPs

We have so far focused on design and analysis of NIMAPs that are secure against a computationally bound adversary. The alternative is to consider adversaries who have access to unbounded amounts of time and resources. In this section, we show that the only NIMAPs which are secure in the presence of such unbounded adversaries are trivial protocols. In other words, the entire message has to be sent over the authenticated channel in order for a NIMAP to be unconditionally secure. In other words, non-trivial NIMAPs that are unconditionally secure do not exist. This result was first proved by Wang and Safavi-Naini [WSN08] using probability distribution arguments. We provide a new proof in the form of a simple counting argument.

2.4.1 Wang and Safavi-Naini's Proof

Wang and Safavi-Naini [WSN08] first showed the impossibility of designing non-trivial unconditionally secure NIMAPs. They used the following model to describe the unconditionally secure NIMAP:

The information theoretic NIMAP model: The sender S sends the message m and some x over the insecure public channel, and a tag t over the manual channel. The receiver R decides whether or not to accept m as authentic from S.

Wang and Safavi-Naini showed that unconditionally secure NIMAPs do not exist without secrets between sender and receiver, and without requirements such as stall-free on the narrow-band channel, unless the whole message is transmitted over the narrow-band channel. This results in a trivial protocol where the authenticated channel has enough bandwidth to transmit the whole message.

They suppose $|m| > |t|$ and propose an attack. First, they show that there definitely exists some other message m' such that m' can be authenticated under some x', possibly different from x, and the same tag t. Now, the adversary, on observing the authentication transcripts (m, x, t), replaces m and x with m' and x'.

They further note that the adversary can mount this attack online by removing m and x from the broadband channel and delaying t on the narrow-band channel until she finds an appropriate m' and x'. Then, she sends m' and x' over the broadband channel and let t be transmitted over the narrow-band channel right after.

In order to formally prove the effectiveness of their attack, for example when proving the existence of appropriate m' and x', they use probability distribution arguments involving Shannon entropies.

2.4.2 A Counting Argument

We now present a much shorter and simpler proof of non-existence of nontrivial NIMAPs. Our proof is based on a counting argument.

We use the same model used by Wang and Safavi-Naini [WSN08] and define $\mathcal{M}$ to be the set of all possible messages to be authenticated and $\mathcal{R}$ to be the set of all possible strings that could be sent on the first flow along with a possible message. Moreover, we let $\mathcal{S}$ be the set of all authenticating tags that are sent over the authenticated channel. An instance of an NIMAP in this model is as follows. A message $M \in \mathcal{M}$ is to be authenticated and it is sent over the broadband channel along with some information $r \in \mathcal{R}$. Later, an authenticating tag $s \in \mathcal{S}$ is sent over the narrow-band channel. Figure 2.16 depicts this NIMAP.

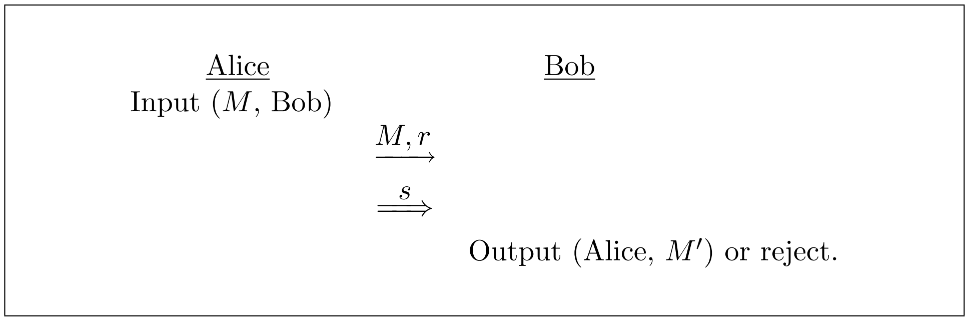

Alice
Bob
Input (M, Bob)
$\xrightarrow{M, r}$
$\overset{s}{\Longrightarrow}$
Output (Alice, M') or reject.

Figure 2.16: A General NIMAP

Let $\mathcal{V}$ be set of all transcripts which result in Bob accepting a message, that is

$$\mathcal{V} = \{(M, r, s) : \text{ Bob accepts the triple } (M, r, s)\}.$$

Note that, $\mathcal{V}$ is public knowledge and a computationally unbounded adversary can find or store $\mathcal{V}$ ahead of time.

If $|\mathcal{M}| \leq |\mathcal{S}|$, then there exists a trivial NIMAP where the whole message is transmitted over the authenticated channel. We assume that $|\mathcal{M}| > |\mathcal{S}|$ to consider non-trivial NIMAPs. For every tag $s \in \mathcal{S}$, we let $\mathcal{M}_s$ be the set of all messages such that there exists some r in which (M, r, s) results in an acceptance by Bob. In other words,

$$\mathcal{M}_s := \{M : (M, r, s) \in \mathcal{V} \text{ for some } r\}.$$

Moreover, we let $\mathcal{U}$ be the set of all tags that can authenticate only one message, that is

$$\mathcal{U} := \{s : |\mathcal{M}_s| = 1\}.$$

Note, $|\mathcal{U}| \leq |\mathcal{S}|$. Next, suppose $|\mathcal{U}| = |\mathcal{S}|$. Then $\sum |\mathcal{M}_s| = |\mathcal{S}|$, which implies $\sum |\mathcal{M}_s| < |\mathcal{M}|$ which is a contradiction. Hence, $|\mathcal{U}| < |\mathcal{S}|$.

Furthermore, we let $\mathcal{M}_{\mathcal{U}}$ be the union of all $\mathcal{M}_s$ such that $s \in \mathcal{U}$. In other words,

$$\mathcal{M}_{\mathcal{U}} = \bigcup_{s \in \mathcal{U}} \mathcal{M}_s.$$

Since $|\mathcal{U}| < |\mathcal{S}|$, we obtain that $|\mathcal{M}_{\mathcal{U}}| < |\mathcal{S}|$, which implies $|\mathcal{M}_{\mathcal{U}}| < |\mathcal{M}|$. Hence, there exists an M in $\mathcal{M} \setminus \mathcal{M}_{\mathcal{U}}$ such that, for any $(M, r, s) \in \mathcal{V}$, there exists $(M', r', s) \in \mathcal{V}$ with $M \neq M'$.

The attack is comprised of Eve choosing any $M \in \mathcal{M} \setminus \mathcal{M}_{\mathcal{U}}$ and giving it to Alice. Later, when she receives (M, r, s) from Alice, she replaces it with the appropriate (M', r', s), that we know exists. Note that Eve is computationally unbounded and can find such an M. Moreover, after receiving (M, r, s) from Alice, Eve finds (M', r'). The attack is depicted in Fig. 2.17.

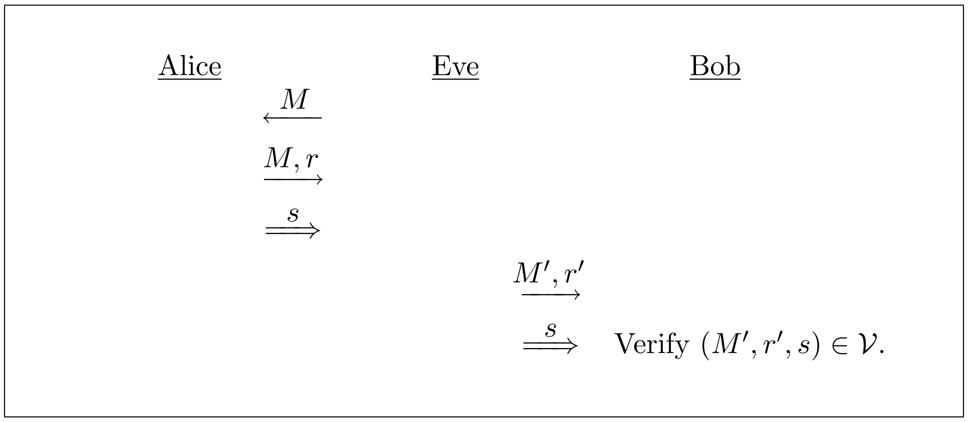


Figure 2.17: An Attack Against the General NIMAP

To conclude, we have shown that non-trivial NIMAPs that are unconditionally secure do not exist.

Chapter 3

Interactive Message Authentication Protocols

Contents

In this chapter, we propose an Interactive Message Authentication Protocol (IMAP) using two channels: an insecure broadband channel and an authenticated narrow-band channel [MS08a]. We consider the problem in the context of ad hoc networks, where it is assumed that there is neither a secret key shared between the two parties, nor a public-key infrastructure in place. The security of our IMAP is based on the existence of Interactive-Collision Resistant (ICR) hash functions, a new notion of hash function security introduced in Section 3.3.1.

In Section 3.1, we summarize the results on existing IMAPs in the literature. In Section 3.2, the attack model is described. Section 3.3 is devoted to introducing our IMAP which is based on the computational assumption that ICR hash functions exist. It performs better than any other message authentication protocols that are based on computational assumptions. That is, while achieving the same level of security, the amount of information sent over the authenticated channel in our IMAP is smaller than the most secure IMAP and Non-interactive Message Authentication Protocol (NIMAP) in the literature. Alternatively, if we send the same amount of information over the authenticated channel, we can allow much stronger adversaries compared to the existing protocols in the literature.

Our protocol has a very simple structure and does not require any long strings to be distributed ahead of time. The security of our protocol is investigated in Section 3.3.3. We allow offline attacks by an adversary. As before, the attack model is the adaptive chosen plain-text attack (ACPA) model. Both substitution and impersonation attacks are analyzed in this model. The ACPA model is a strong model, and as a result, a scheme that is proven secure in this model does not require authenticated channels that have any unusual properties. In the ACPA model, the adversary has offline computational power and can induce the users to send messages of the adversary's choice. In this chapter, we give further power to the adversaries by allowing them to have online computational power. That is, they are allowed to do hash function computations, or make oracle queries, while they are in the middle of an attack.

Furthermore, our IMAP benefits from a simple structure and works under fewer security assumptions compared to other IMAPs in the literature. The simplicity and the easy-to-use structure of the protocol makes it applicable in a wide variety of real-world settings where ad hoc networks have no trusted infrastructure. For instance, it can be used in pairings of wireless devices such as Wireless USB and Bluetooth, in Personal Area Networks (PANs), or in a disaster case where a trusted infrastructure has been compromised.

We analyze the security and efficiency of our IMAP and show that the performance of our IMAP is better than that of other IMAPs and NIMAPs proposed so far. In other words, our IMAP achieves a better level of security, while benefiting from an efficient structure and having to send fewer bits over the authenticated channel. To reiterate, if we want to send the same amount of information, then we can assume much stronger adversaries in terms of online computational complexity.

This chapter is concluded in Section 3.4 by proposing a generalization of an unconditionally secure IMAP [NSS06]. Sufficient conditions for our IMAP to be secure are found.

3.1 Previous IMAPs

A non-interactive protocol is, in general, preferred to an interactive protocol if they are achieving the exact same goals. In other words, interactive protocols are supposed to either achieve better security or be more efficient than their non-interactive competitors, otherwise, one would choose to implement non-interactive protocols and obtain the same results. For instance, having a bidirectional channel may cost more than a unidirectional channel, or devices may have different computational capabilities, allowing one device to be the master and the other be the slave in the communication. However, we note that NIMAPs achieve a strictly weaker notion of security when compared to IMAPs. This is because NIMAPs, on their own, provably cannot protect against replay attacks of the authenticated flow, while IMAPs can. (One could use a time-stamping technique to solve this problem for NIMAPs, however.)

The IMAP presented in this chapter is based on a computational assumption. As a result, we can only compare its security and efficiency to similar IMAPs that are based on computational assumptions.

Hoepman [Hoe04] proposed an authenticated key agreement protocol that uses both a bidirectional narrow-band channel and a bidirectional broadband channel. This interactive protocol consists of a commitment exchange, an authentication exchange, and finally a decisional Diffie-Hellman problem in a group G. The security is based on the hardness of the decisional Diffie-Hellman problem in G and on two hash functions H_1 and H_2 having a very specific structure. Vaudenay [Vau05] observed that instances of such hash functions may not exist at all.

Vaudenay [Vau05] proposed an IMAP based on equivocable or extractable commitment schemes. The protocol is designed in the Common Reference String model where a random string is authentically predistributed among all users. The random string is then used to compute commitment values. This protocol is depicted in Fig. 3.1.

By targeting typical parameters, such as offline complexity of 2^{70} and $q = 2^{10}$, Vaudenay's protocol requires authenticating 50 bits to get the probability of success of the adversary to be at most 2^{-20}. This protocol achieves a good level of security when compared to other proposals in the literature. However, the only efficient commitment schemes, with the specific properties required here, are in the Random Oracle Model. There are other instances of such commitment schemes in the standard model, but the number of rounds is logarithmic in terms of the secu-

Alice		Bob
Input (Bob, M), Choose $R_A \in \{0,1\}^k$ uniformly at random		Choose $R_B \in \{0,1\}^k$ uniformly at random
$(c,d) \leftarrow commit(M, R_A)$	$\xrightarrow{(M \Vert c)}$	Receive (M', c')
Receive R'_B	$\xleftarrow{R_B}$	
	$\xrightarrow{d}$	Receive d' and compute $R'_A \leftarrow open(M', c', d')$
Compute $R = R_A \bigoplus R'_B$	$\xRightarrow{R}$	If $R = R'_A \bigoplus R_B$, then output (Alice, M') and reject otherwise.

Figure 3.1: Vaudenay's IMAP

rity parameters and their analyses involve zero-knowledge proofs. Also, there are some efficient commitment schemes with the appropriate properties in the Common Random String (CRS) model. However, the CRS model might not be suitable in an ad hoc setting where it is not practical to authentically distribute a random string to every user. We note that the possibility that the adversary does online computations has not been considered in Vaudenay's protocol.

Another recent paper, by Naor, Segev and Smith [NSS06], investigates two-channel authentication in the interactive setting. They achieve unconditional security using evaluation of polynomials over finite fields. For every integer k, their IMAP allows the sender to authenticate an n-bit message in k rounds, such that the length of the authenticated string is about $2\log(1/\epsilon)+2\log^{(k-1)} n+O(1)$. By setting $k = \log(n) + O(1)$, the manually authenticated string is of length $2\log(1/\epsilon)$. They conclude that the advantage of assuming computational security is to reduce the amount of information that needs to be authenticated from $2\log(1/\epsilon)$ to $\log(1/\epsilon)$, and not to reduce the number of flows of the protocol.

3.2 The Attack Model

The adversary is trying to make Bob accept a message M' along with the identity of Alice, when in fact the message M' was never sent by Alice to Bob. That is, the *adversarial goal* is to make Bob output (Alice, M') when he was supposed to reject (Alice, M') or accept (Alice, M) for some s$M \neq M'$. There are two main types of attacks to consider: *impersonation* attacks and *substitution* attacks.

In an impersonation attack, the adversary initiates a session and tries to convince Bob that a message M' is sent from Alice, while in fact M' was never sent from Alice. In our model, the attacker cannot initiate a new authenticated flow. She can only replay a previous authenticated flow. Hence, the authenticated flow in an impersonation attack constitutes of a replay of a previous authenticated flow sent by Alice to Bob.

On the other hand, a substitution attack occurs when Alice initiates a session with Bob, and tries to send him a message M. Then, the attacker substitutes M' for M, so, Bob receives M' and not M. The authenticated flow cannot be substituted according to the model, and hence any potential changes occur in the broadband channel. There are two types of substitution attacks; see Section 3.3.3.

Moreover, we assume that the adversary can make Alice send a message that the adversary has chosen. This ability of the adversary may not be considered in all models. We do consider it in our model since it makes the adversary more powerful and results in a stronger level of security. The adaptive chosen plaintext attack (ACPA) model is very strong and desirable compared to other models. It consists of two stages: an *information gathering* stage and a *deception* stage. In addition, we assume that the attacker has precomputing capabilities and is able to mount "dictionary-type" attacks.

The term *offline complexity* is used to refer to the computational complexity $T_{\text{off}} = 2^{t_{\text{off}}}$ of an adversary up to and including the information gathering stage. The term *online complexity* refers to the computational complexity $T_{\text{on}} = 2^{t_{\text{on}}}$ of an adversary during the deception stage of a substitution attack. Furthermore, the number of messages sent by Alice to Bob during the information gathering stage is denoted by q. The parameter t_{off} is chosen in agreement with the usual capabilities of a computationally bounded adversary assuming today's computational power of computers. For instance, $t_{\text{off}} \leq 80$ or $t_{\text{off}} \leq 70$ are commonly used bounds in the literature as of December of 2008. The choice for the parameter t_{on} depends on the structure and application scenario of the particular protocol under discussion.

In the information gathering stage, the adversary is allowed to adaptively choose q messages and make Alice send them to Bob. The communication is then recorded for further use. The adversary hopes that this stage of an attack gradually reveals information about the unknown aspects of the protocol.

The deception stage happens after the information gathering phase. The attacker tries to make Bob accept a message M' along with the identity of Alice, when he was supposed to reject. We note that the message M' should be different from all the messages previously sent by Alice, otherwise we consider the "attack" only a "replay".

Let $\mathcal{M}$ be the set of all messages, and let $x \in \mathcal{X}$, $y \in \mathcal{Y}$, and $s \in \mathcal{S}$, for some sets $\mathcal{X}, \mathcal{Y}$, and $\mathcal{S}$. Figure 3.2 depicts a 3-round generic IMAP (3GIMAP).

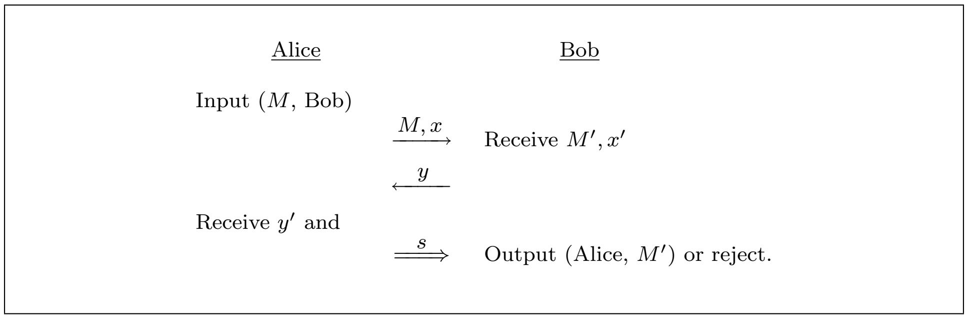


Figure 3.2: 3GIMAP

Gehrman [Geh98] looked at different possible attacks against a generic k-round protocol and proved that there are in total $\binom{k+1}{\frac{k+1}{2}}$ distinct attacks. He used the following notation to label these attacks. A flow initiated by the adversary is labelled as **A** if it sent to Alice, and, similarly, a flow sent by the adversary is labelled as **B** if the recipient is Bob. According to his result, there are $\binom{4}{2} = 6$ possible attacks against a three round protocol, namely AABB, ABBA, BABA, ABAB, BBAA, and BAAB attacks.

The last flow of 3GIMAP is an authenticated flow sent by Alice to Bob. According to the communication model of two-channel cryptography, the adversary can only replay this last flow. As a result, the only possible attacks against 3GIMAP are the ones that end with a flow sent to Bob, namely AABB, ABAB, and BAAB. These attacks are depicted in Figures 3.4, 3.5, and 3.3.

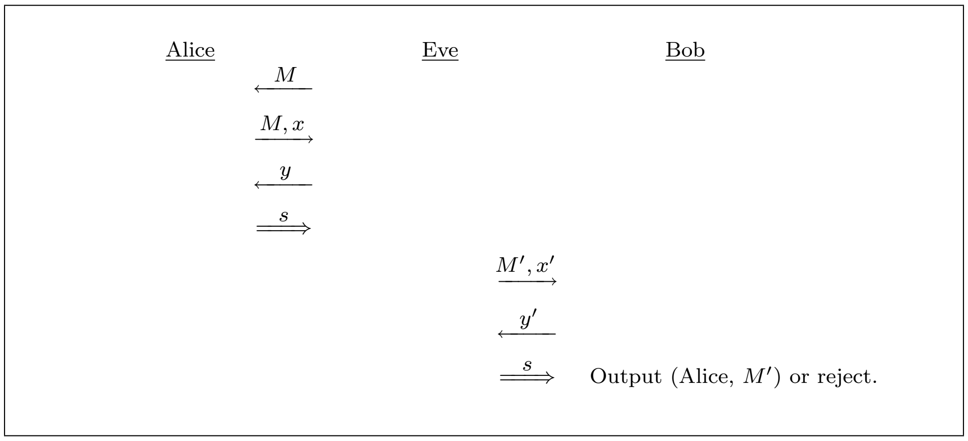


Figure 3.3: Attack of Type AABB

The attack of type AABB is an impersonation attack whereas the attacks of type BAAB and type ABAB are substitution attacks. Details are explained in

Section 3.3.3.

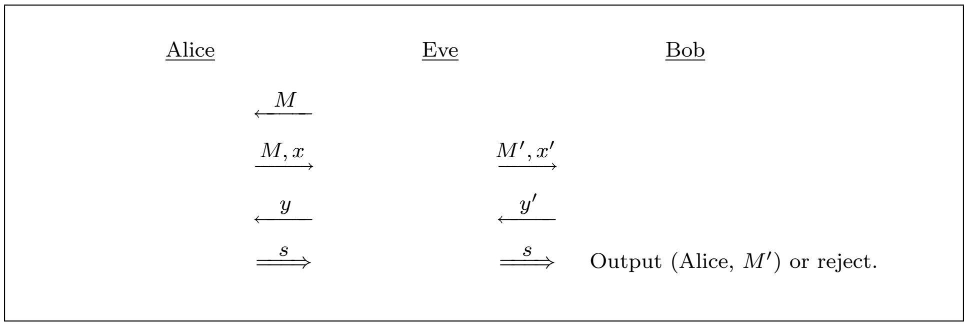


Figure 3.4: Attack of Type ABAB

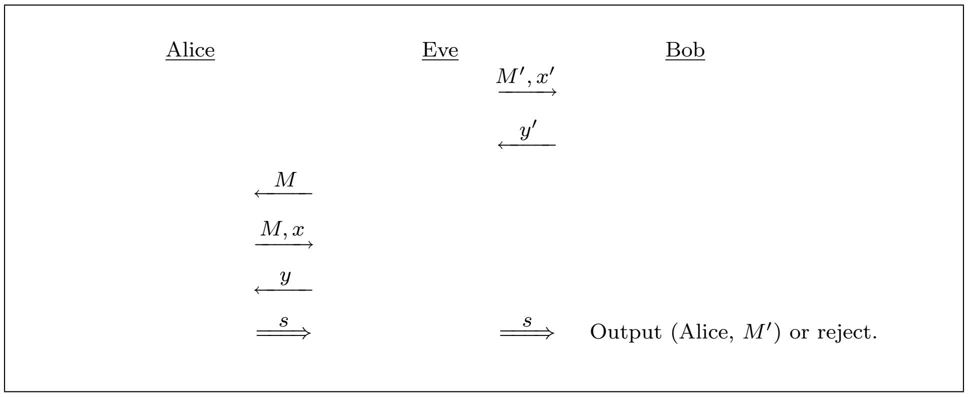


Figure 3.5: Attack of Type BAAB

3.3 A New Computationally Secure IMAP

We begin by defining new notion of hash function security which we call Interactive-Collision Resistance (ICR). We continue by introducing a new IMAP based on ICR hash functions. The security of this IMAP is based on the hardness of the ICR problems.

3.3.1 Interactive-Collision Resistance

In this section, we begin by defining Interactive-Collision Resistance I, II and III (ICRI, ICRII, and ICRIII respectively) for hash functions. Then, we state and prove three lemmas about the security of ICRI, ICRII, and ICRIII hash functions. ICRI, ICRII, and ICRIII properties are going to guard our protocol against attack of type ABAB, BAAB, and AABB, respectively.

To our knowledge, this is the first time that the problem of finding interactive-collisions of type I, II, and III are being investigated. We analyze the ICRI, ICRII, and ICRIII Games in the Random Oracle Model. This analysis yields some insight into the hardness of these games compared to Collision Resistance (CR) or Second-Preimage Resistance (SPR). Note that we do not have any concrete constructions for designing such hash functions in the standard model. We pose this as an open problem.

Definition 1. *A hash function H is* **Interactive-Collision Resistant** I **(ICR**I**)** *if the game of Fig. 3.6 is hard to win, for fixed values of ℓ_1, ℓ_2, and ℓ_3. In addition, the pair $(M\|K\|R', M'\|K'\|R)$ is called an interactive-collision of type* I*. Furthermore, we call H a $(T_{\text{off}}, \epsilon_1)$-ICR*I *hash function if an adversary, who can make up to T_{off} hash function computations, wins the ICR*I *game with probability at most ϵ_1.*

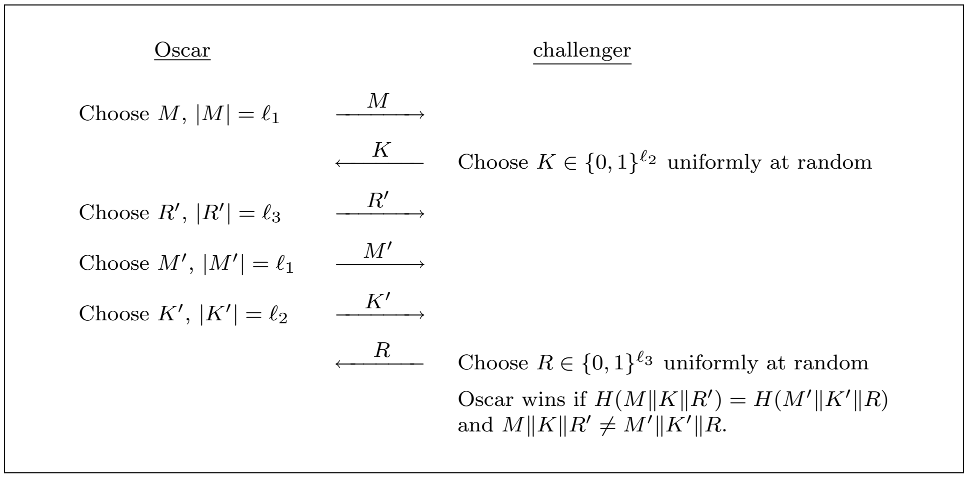

Oscar
challenger
Choose M, $|M| = \ell_1$
M
K
Choose $K \in \{0,1\}^{\ell_2}$ uniformly at random
Choose R', $|R'| = \ell_3$
R'
Choose M', $|M'| = \ell_1$
M'
Choose K', $|K'| = \ell_2$
K'
R
Choose $R \in \{0,1\}^{\ell_3}$ uniformly at random
Oscar wins if $H(M\|K\|R') = H(M'\|K'\|R)$ and $M\|K\|R' \neq M'\|K'\|R$.

Figure 3.6: ICRI Game

Note that if $\ell_2 = \ell_3 = 0$, then ICRI is equivalent to Collision Resistance (CR). Further, if $\ell_1 = \ell_3 = 0$, then ICRI is equivalent to Second-Preimage Resistance (SPR). In fact, ICRI is interpolating between CR and SPR. This suggests that solving ICRI game is at least as hard as finding collisions, but no harder than finding second-preimages.

We can analyze the security of ICRI hash functions, or in other words the hardness of the ICRI game, in the Random Oracle Model. This will give us an intuition on how difficult this game is, as compared to former notions of hash function security. Let $\mathcal{F}^{\mathcal{X},\mathcal{Y}}$ denote the set of all functions from a domain $\mathcal{X}$ to a range $\mathcal{Y}$.

Lemma 1. *Let $\mathcal{X} = \{0,1\}^{\ell_1+\ell_2+\ell_3}$ be the set of all possible binary strings of length $\ell_1+\ell_2+\ell_3$. Consider a hash function H chosen randomly from $\mathcal{F}^{\mathcal{X},\mathcal{Y}}$, where $|\mathcal{Y}| = 2^k$. Then, H is a $(2^{t_{\text{off}}}, \epsilon_1)$-ICR*I *hash function in the Random Oracle model, where $\epsilon_1 = 2^{-k}(1 + 2^{2t_{\text{off}}-\ell_2-\ell_3} + 2^{t_{\text{off}}-\ell_3})$. In other words, any player with computational complexity $T_{\text{off}} = 2^{t_{\text{off}}}$ against the challenger of the ICR*I *game has a probability of success at most $\epsilon_1 = 2^{-k}(1 + 2^{2t_{\text{off}}-\ell_2-\ell_3} + 2^{t_{\text{off}}-\ell_3})$.*

Proof. We consider $\mathcal{X} = \{0,1\}^{\ell_1+\ell_2+\ell_3}$, the set of all possible binary strings of size $\ell_1+\ell_2+\ell_3$, and let a hash function H be chosen randomly from $\mathcal{F}^{\mathcal{X},\mathcal{Y}}$, where $|\mathcal{Y}| = 2^k$.

Assume that we are only permitted oracle access to H, that is we are working in the Random Oracle Model. We let the adversary have access to the Random Oracle for $T_{\text{off}} = 2^{t_{\text{off}}}$ times. Given these conditions, we are looking for the probability ϵ_1 of Oscar winning the ICRI game.

Let $\mathcal{A} = \{X_1, X_2, \ldots, X_{T_{\text{off}}}\}$ be the queries of Oscar to the Random Oracle, where $|X_i| = \ell_1+\ell_2+\ell_3$ for $1 \le i \le T_{\text{off}}$. Without loss of generality, we assume that X_is are distinct, for $1 \le i \le T_{\text{off}} = 2^{t_{\text{off}}}$.

Consider the pair $(Y, Y') = (M\|K\|R', M'\|K'\|R)$ and write X_is in the form of $X_i = M_i\|K_i\|R_i$, where $|M_i| = \ell_1$, $|K_i| = \ell_2$ and $|R_i| = \ell_3$.

We want to find an upper bound on the probability of Oscar winning the ICRI game by finding Y and Y' such that $H(Y) = H(Y')$. We will do this by considering the following cases:

Case 1. $Y' \notin \mathcal{A}$, that is, Y' is not a precomputed value;

Case 2. $Y' \in \mathcal{A}$ and $Y \notin \mathcal{A}$, that is, Y' is precomputed, but Y is not;

Case 3. $Y' \in \mathcal{A}$ and $Y \in \mathcal{A}$, that is, Y and Y' are both precomputed.

Note that these three cases are mutually exclusive and they cover all possibilities. We will discuss each case separately.

Case 1. Notice that Y is determined after the third flow. Moreover, Y' is not a precomputed value and yet it collides with Y. Furthermore, Y was determined before Y' was chosen. In this case, the probability that $H(Y) = H(Y')$ is 2^{-k} due to the properties of Random Oracles.

Case 2. In this case, Y' is precomputed and Y is not. After the third flow, when Y is determined, the probability that $H(Y) = H(X_j)$, for some j, is $2^{t_{\text{off}}-k}$. Now, Oscar wants $Y' = X_j$. Therefore, he sends $M' = M_j$ and $K' = K_j$. Then, the challenger responds with a random R'. As a result,

$$\Pr[Y' = X_j] = \Pr[R' = R_j] = 2^{-\ell_3}.$$

Hence, the probability that $H(Y) = H(Y')$ and $Y' = X_j$, when Y is not a precomputed value, is $2^{t_{\text{off}}-k-\ell_3}$.

Case 3. When both Y and Y' are precomputed values, it means that Oscar had found a collision among the precomputed values. Let the colliding values be $X_i = M_i \| K_i \| R_i$ and $X_j = M_j \| K_j \| R_j$, $1 \leq i, j \leq 2^{t_{\text{off}}}$ with $i \neq j$. We know that the probability of finding a collision among T_{off} random values is $\binom{T_{\text{off}}}{2}/2^k$. This is approximately equal to $2^{2t_{\text{off}}-k-1}$ when $T_{\text{off}} = 2^{t_{\text{off}}}$.

After finding a collision in $\mathcal{A}$, Oscar wants $Y = X_i$ and $Y' = X_j$. Therefore, he lets $M = M_i, R' = R_i, M' = M_j$, and $K' = K_j$. Note that K and R are being chosen by the challenger. The probability of a random K being equal to a precomputed K_i is $2^{-\ell_2}$. Similarly, the probability of a random R being equal to a precomputed R_j is $2^{-\ell_3}$. We obtain

$$\Pr[(Y, Y') = (X_i, X_j)] = \Pr[K = K_i] \times \Pr[R = R_j] = 2^{-\ell_2-\ell_3}.$$

As a result, we get

$$\Pr[H(X_i) = H(X_j) \text{ and } \{Y, Y'\} = \{X_i, X_j\}] = 2^{2t_{\text{off}}-k-\ell_2-\ell_3}.$$

Considering all three cases, we conclude that

$$\Pr[H(Y) = H(Y')] = 2^{-k}(1 + 2^{2t_{\text{off}}-\ell_2-\ell_3} + 2^{t_{\text{off}}-\ell_3}).$$

□

The above discussion concludes the proof of Lemma 1. To reiterate the Lemma, one can say that any player with computational complexity $T_{\text{off}} = 2^{t_{\text{off}}}$ against the challenger of the ICRI game has a probability of success at most $\epsilon_1 = 2^{-k}(1 + 2^{2t_{\text{off}}-\ell_2-\ell_3} + 2^{t_{\text{off}}-\ell_3})$.

We now define Interactive-Collision Resistance II.

Definition 2. *A hash function H is* **Interactive-Collision Resistant** II **(ICR**II**)** *if the game of Fig. 3.7 is hard to win, for fixed values of ℓ_1, ℓ_2, and ℓ_3. The pair $(M\|K\|R', M'\|K'\|R)$ is called an interactive-collision of type* II*. Furthermore, we call H a $(T_{\text{off}}, T_{\text{on}}, \epsilon_2)$-ICR*II *hash function if an adversary with offline complexity T_{off} and online complexity T_{on} wins the ICR*II *game with probability at most ϵ_2.*

<u>Oscar</u>		<u>challenger</u>
Choose M, $\|M\| = \ell_1$	$\xrightarrow{M}$	
	$\xleftarrow{K}$	Choose $K \in \{0,1\}^{\ell_2}$ uniformly at random
Choose M', $\|M'\| = \ell_1$	$\xrightarrow{M'}$	
Choose K', $\|K'\| = \ell_2$	$\xrightarrow{K'}$	
	$\xleftarrow{R}$	Choose $R \in \{0,1\}^{\ell_3}$ uniformly at random
Choose R', $\|R'\| = \ell_3$	$\xrightarrow{R'}$	
		Oscar wins if $H(M\|K\|R') = H(M'\|K'\|R)$ and $M\|K\|R' \neq M'\|K'\|R$.

Figure 3.7: ICRII Game

As in ICRI, if $\ell_2 = \ell_3 = 0$, then ICRII is equivalent to Collision Resistance. As a result, we conclude that finding collisions is no harder than finding interactive-collisions of type II. On the other hand, if $\ell_1 = \ell_2 = 0$, then ICRII is very close to Second-Preimage Resistance (the lengths are fixed in the ICRII notion).

Similar to ICRI, we analyze the security of ICRII hash functions in the Random Oracle Model to have an intuition on how difficult it is to win the ICRII game.

Lemma 2. *Let $\mathcal{X} = \{0,1\}^{\ell_1+\ell_2+\ell_3}$ be the set of all possible binary strings of size $\ell_1 + \ell_2 + \ell_3$. Consider a hash function H chosen randomly from $\mathcal{F}^{\mathcal{X},\mathcal{Y}}$, where $|\mathcal{Y}| = 2^k$. Then, H is a $(2^{t_{\text{off}}}, 2^{t_{\text{on}}}, \epsilon_2)$-ICR*II *hash function in the Random Oracle Model, where $\epsilon_2 = 2^{-k}(1 + 2^{2t_{\text{off}}-\ell_2-\ell_3} + 2^{t_{\text{off}}-\ell_3} + 2^{t_{\text{on}}})$. In other words, any player with offline computational complexity $T_{\text{off}} = 2^{t_{\text{off}}}$ and online complexity $T_{\text{on}} = 2^{t_{\text{on}}}$ against the challenger of the ICR*II *game has a probability of success at most $\epsilon_2 = 2^{-k}(1 + 2^{2t_{\text{off}}-\ell_2-\ell_3} + 2^{t_{\text{off}}-\ell_2} + 2^{t_{\text{on}}})$.*

Proof of Lemma 2. Let H again be a Random Oracle and assume that the adversary can access the Random Oracle for up to $T_{\text{off}} = 2^{t_{\text{off}}}$ times before he receives the last flow from the challenger, i.e. the value of R in the ICRII. Furthermore, he can access the Random Oracle for up to $T_{\text{on}} = 2^{t_{\text{on}}}$ times after he receives the last flow from the challenger and before he sends the value of R'. We now find an upper bound on the probability ϵ_2 of Oscar winning the ICRII game.

Let the pair $(Y, Y') = (M\|K\|R', M'\|K'\|R)$ be the interactive-collision of type II found by Oscar. Further, let $\mathcal{A} = \{X_1, \ldots, X_{T_{\text{off}}}\}$ be Oscar's inputs to the Random Oracle before he receives the value of R from the challenger, and $\mathcal{B} = \{X_{T_{\text{off}}+1}, \ldots, X_{T_{\text{off}}+T_{\text{on}}}\}$ be his inputs to the Random Oracle after he received the value of R. Without loss of generality, we assume that $X_1, \ldots, X_{T_{\text{off}}+T_{\text{on}}}$ are all distinct. We write each X_i in the form of $M_i\|K_i\|R_i$, where $|M_i| = \ell_1$, $|K_i| = \ell_2$ and $|R_i| = \ell_3$.

We would like to find an upper bound on the probability of Oscar winning the ICRII game. We will do this by considering the following mutually exclusive cases:

Case 1. $Y \in \mathcal{B}$, that is, Y is one of the T_{on} values computed after the fifth flow;

Case 2. $Y \notin \mathcal{B}$ and $Y \notin \mathcal{A}$, that is, Y is not among the T_{off} precomputed values or the T_{on} values computed after the fifth flow;

Case 3. $Y \in \mathcal{A}$, and $Y' \notin \mathcal{A}$, that is, Y is among the T_{off} precomputed values and Y' is not.

Case 4. $Y \in \mathcal{A}$, and $Y' \in \mathcal{A}$, that is, Y and Y' are both among the T_{off} precomputed values.

Note that these mutually exclusive cases cover all possibilities. We now treat each case separately.

Case 1. Once the challenger sends R, Y' is determined. Hence, the probability that Y' collides with one of the T_{on} values that Oscar computes after receiving R is $2^{t_{\text{on}}-k}$. Oscar chooses T_{on} values that all begin with $M\|K$ (from the first two flows). Then, if a collision is found with Y', Oscar can choose R' so that Y collides with Y'.

Case 2. In this case, Y is neither precomputed among the offline computations and nor is it computed during the online computations. Note that Y' is determined once the challenger sends R. The probability that Y, a value which is not precomputed, collides with a determined value Y' is 2^{-k}.

Case 3. This case refers to the situation where Y is computed among the offline computations and Y' is not. Note that Y' is determined after the fifth flow. The probability that $H(Y') = H(X_j)$, for some $j \leq T_{\text{off}}$, is $2^{t_{\text{off}}-k}$. Now, Oscar wants $Y = X_j$. Therefore, he sends $M = M_j$ and $R' = R_j$. We note that K is chosen by the challenger. As a result,

$$\Pr[Y = X_j] = \Pr[K = K_j] = 2^{-\ell_2}.$$

Hence, the probability that $H(X_j) = H(Y')$ and $Y = X_j$, when Y' is not a precomputed value, is $2^{t_{\text{off}}-k-\ell_2}$.

Case 4. If Y and Y' are both precomputed during the offline computations, then Oscar has found a collision before starting the game. Let the colliding values be denoted by $X_i = M_i \| K_i \| R_i$ and $X_j = M_j \| K_j \| R_j$, $1 \leq i, j \leq 2^{t_{\text{off}}}$ with $i \neq j$. The probability of finding a collision among T_{off} random values is $\binom{T_{\text{off}}}{2}/2^k$. This is approximately equal to $2^{2t_{\text{off}}-k-1}$ for $T_{\text{off}} = 2^{t_{\text{off}}}$.

After finding a collision in $\mathcal{A}$, Oscar wants $Y = X_i$ and $Y' = X_j$. Hence, he lets $M = M_i, R' = R_i, M' = M_j$, and $K' = K_j$. Note that K and R are being chosen by the challenger. The probability of a random K being equal to a precomputed K_i is $2^{-\ell_2}$. Similarly, the probability of a random R being equal to a precomputed R_j is $2^{-\ell_3}$. As a result, we obtain

$$\Pr[(Y, Y') = (X_i, X_j)] = \Pr[K = K_i] \times \Pr[R = R_j] = 2^{-\ell_2-\ell_3}.$$

As a result, we get

$$\Pr[H(X_i) = H(X_j) \text{ and } \{Y, Y'\} = \{X_i, X_j\}] = 2^{2t_{\text{off}}-k-\ell_2-\ell_3}.$$

Summing up the above cases, we obtain that

$$\Pr[H(Y) = H(Y')] = 2^{-k}(1 + 2^{2t_{\text{off}}-\ell_2-\ell_3} + 2^{t_{\text{off}}-\ell_2} + 2^{t_{\text{on}}})$$

□

Next, we define Interactive-Collision Resistant III (ICRIII).

Definition 3. *A hash function H is* **Interactive-Collision Resistant** III **(ICR**III**)** *if the game of Fig. 3.8 is hard to win, for fixed values of ℓ_1, ℓ_2, and ℓ_3. The pair $(M\|K\|R', M'\|K'\|R)$ is called an interactive-collision of type* III. *Furthermore, we call H a $(T_{\text{off}}, T_{\text{on}}, \epsilon_3)$-ICR*III *hash function if an adversary with offline complexity T_{off} and online complexity T_{on} wins the ICR*III *game with probability at most ϵ_3.*

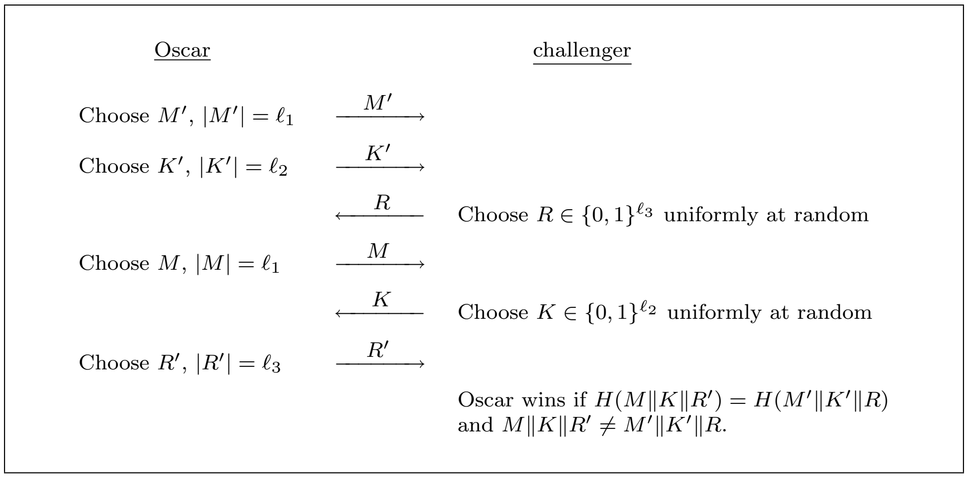

Oscar
challenger
Choose M', $|M'| = \ell_1$ $\xrightarrow{M'}$
Choose K', $|K'| = \ell_2$ $\xrightarrow{K'}$
$\xleftarrow{R}$ Choose $R \in \{0,1\}^{\ell_3}$ uniformly at random
Choose M, $|M| = \ell_1$ $\xrightarrow{M}$
$\xleftarrow{K}$ Choose $K \in \{0,1\}^{\ell_2}$ uniformly at random
Choose R', $|R'| = \ell_3$ $\xrightarrow{R'}$
Oscar wins if $H(M\|K\|R') = H(M'\|K'\|R)$ and $M\|K\|R' \neq M'\|K'\|R$.

Figure 3.8: ICRIII Game

Lemma 3. *Let $\mathcal{X} = \{0,1\}^{\ell_1+\ell_2+\ell_3}$ be the set of all possible binary strings of size $\ell_1 + \ell_2 + \ell_3$. Consider a hash function H chosen randomly from $\mathcal{F}^{\mathcal{X},\mathcal{Y}}$, where $|\mathcal{Y}| = 2^k$. Then, H is a $(2^{t_{\text{off}}}, 2^{t_{\text{on}}}, \epsilon_3)$-ICR*III *hash function in the Random Oracle Model, where $\epsilon_3 = 2^{-k}(1 + 2^{2t_{\text{off}}-\ell_2-\ell_3} + 2^{t_{\text{off}}-\ell_3} + 2^{t_{\text{on}}})$. In other words, any player with offline computational complexity $T_{\text{off}} = 2^{t_{\text{off}}}$ and online complexity $T_{\text{on}} = 2^{t_{\text{on}}}$ against the challenger of the ICR*III *game has a probability of success at most $\epsilon_3 = 2^{-k}(1 + 2^{2t_{\text{off}}-\ell_2-\ell_3} + 2^{t_{\text{off}}-\ell_2} + 2^{t_{\text{on}}})$.*

The proof of Lemma 3 is similar to the proof of Lemma 2.

Finally, we define the notion of an Interactive-Collision Resistant hash function.

Definition 4. *A hash function H is* **Interactive-Collision Resistant (ICR)** *if the ICR*I, *ICR*II, *and ICR*III *games are all hard to win.*

*Furthermore, H is said to be a $(T_{\text{off}}, T_{\text{on}}, \epsilon_1, \epsilon_2)$-ICR hash function if it is a $(T_{\text{off}}, \epsilon_1)$-ICR*I *hash function, a $(T_{\text{off}}, T_{\text{on}}, \epsilon_2)$-ICR*II *hash function, and a $(T_{\text{off}}, T_{\text{on}}, \epsilon_2)$-ICR*III *hash function.*

3.3.2 A New IMAP Using ICR Hash Functions

Let H be a $(T_{\text{off}}, T_{\text{on}}, \epsilon_1, \epsilon_2)$-ICR hash function with fixed parameters ℓ_1, ℓ_2, and ℓ_3. We propose the following IMAP:

1. On input (M, Bob), Alice chooses $K \in \{0,1\}^{\ell_2}$ uniformly at random and sends $M\|K$ to Bob over the insecure channel.

2. Bob receives $M'\|K'$.
3. Bob chooses $R \in \{0,1\}^{\ell_3}$ uniformly at random and he sends it to Alice.
4. Alice receives R'.
5. Alice computes $h = H(M\|K\|R')$ and sends it over the authenticated channel.
5. Bob receives h'.
6. Bob computes $H(M'\|K'\|R)$.
7. Bob outputs (Alice, M') if $h' = H(M'\|K'\|R)$, and he rejects otherwise.

This IMAP is illustrated in Fig. 3.9. Next, we prove that this IMAP is secure under the assumption that the three games in Figures 3.6, 3.7, and 3.8 are hard to win. In other words, if H is a $(T_{\text{off}}, T_{\text{on}}, \epsilon_1, \epsilon_2)$-ICR hash function, then the IMAP is secure.

Alice		Bob
Input (M, Bob)		
Choose $K \in_R \{0,1\}^{\ell_2}$	$\xrightarrow{M\|K}$	Receive $M'\|K'$
	$\xleftarrow{R}$	Choose $R \in_R \{0,1\}^{\ell_3}$
Receive R' and		
Compute $h = H(M\|K\|R')$	$\xRightarrow{h}$	Output (Alice, M') if $h = H(M'\|K'\|R)$, and reject otherwise.

Figure 3.9: Interactive Message Authentication Protocol

3.3.3 Security Analysis

In this section, we analyze the security of the IMAP presented in Fig. 3.9. We consider substitution and impersonation attacks separately. Associated with each attack scenario, an IMAP game is introduced. Winning this game is equivalent to attacking our proposed IMAP. Finally, the reduction of the ICRI, and similarly ICRII and ICRIII, to the IMAP game is shown.

As was mentioned earlier, the ACPA model consists of an information gathering stage and the deception stage.

3.3.3.1 The Information Gathering Stage

During the information gathering stage, the adversary can change the information sent over the broadband channel. For instance, the adversary may change R to R', or K to K'. The other value that is being sent over the broadband channel is the message M. However, our model allows the adversary to choose the message M to start with. Hence, there is no need for the adversary to intervene and change it to M'. Since we are working in the ACPA model, the adversary can make Alice send q messages in the information gathering stage. This stage is depicted in Fig. 3.10.

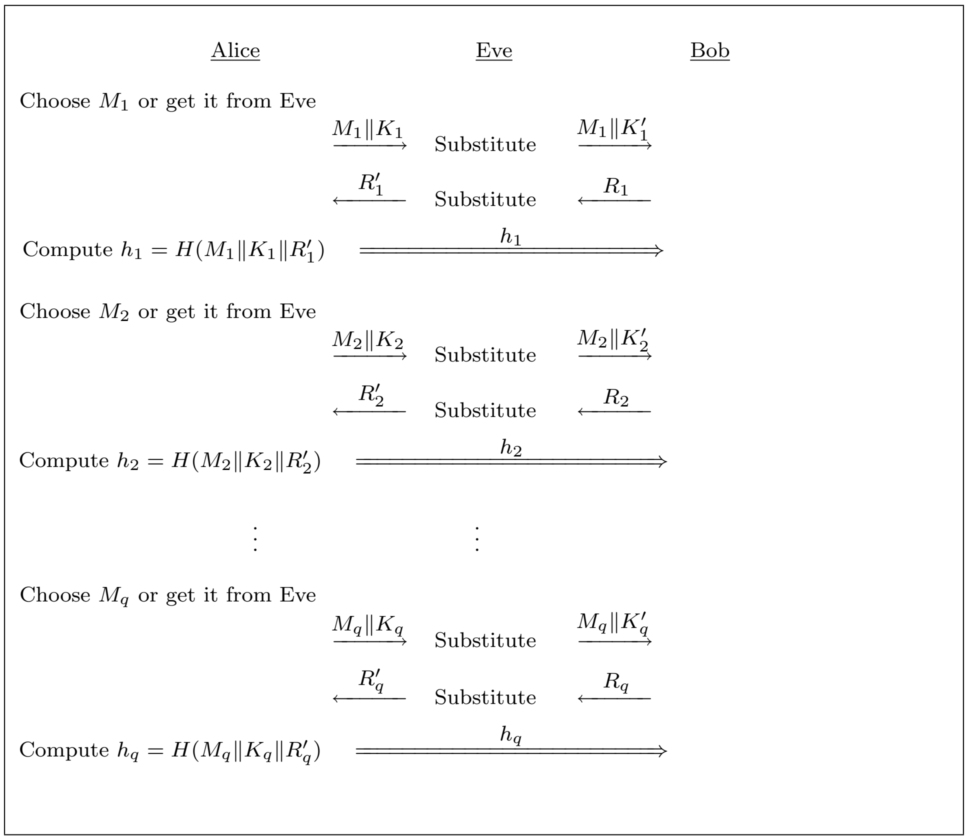

Alice
Eve
Bob
Choose M_1 or get it from Eve
$M_1\|K_1$
Substitute
$M_1\|K'_1$
R'_1
Substitute
R_1
Compute $h_1 = H(M_1\|K_1\|R'_1)$
h_1
Choose M_2 or get it from Eve
$M_2\|K_2$
Substitute
$M_2\|K'_2$
R'_2
Substitute
R_2
Compute $h_2 = H(M_2\|K_2\|R'_2)$
h_2
Choose M_q or get it from Eve
$M_q\|K_q$
Substitute
$M_q\|K'_q$
R'_q
Substitute
R_q
Compute $h_q = H(M_q\|K_q\|R'_q)$
h_q

Figure 3.10: Information Gathering Phase of an Attack

As mentioned previously, the goal of the adversary in attacking a MAP is to make the verifier, Bob, accept a message M' along with the identity of the claimant, Alice, when he was supposed to reject and, indeed, the message M' was never sent by Alice to Bob. There are two main ways of achieving this goal: by mounting impersonation attacks or substitution attacks. We will prove that a successful impersonation attack translates into winning the ICRI game and a successful substitution attack is equivalent to winning either the ICRII game or the ICRIII game.

3.3.3.2 Impersonation Attack

Figure 3.11 depicts the impersonation attack against our IMAP. Here, the attacker initiates a session herself and tries to convince Bob that a message M' is sent from Alice, while in fact M' was generated by the attacker and Alice never sent M' to Bob. This attack is analogous to an attack of type AABB depicted in Fig. 3.3.

Eve		Bob
Input (M', Bob)	$\xrightarrow{M'\Vert K'}$	Receive $M'\Vert K'$
	$\xleftarrow{R'}$	Choose $R' \in \{0,1\}^{\ell_2}$
Receive R'		uniformly at random
Send $h = h_i = H(M_i\Vert K_i\Vert R'_i)$ for	$\xRightarrow{h}$	Output (Alice, M') if $h = H(M'\Vert K'\Vert R')$, for
some i, $1 \leq i \leq q$.		and reject otherwise.

Figure 3.11: An Impersonation Attack Against IMAP

According to our model, the data sent over the authenticated channel, although public, cannot be modified by the adversary. Hence, Eve can only replay a previous flow sent by Alice, as shown in Fig. 3.11. The attacker replays one of $h_1, \ldots, h_q$. Given that Alice has never sent M', the adversarial goal is achieved if Bob accepts.

IMAP Game Against Impersonation Attacks. We now prove that our IMAP is secure against impersonation attacks mounted by an adversary who has offline computational power T_{off} given that H is a $(T_{\text{off}}, \epsilon_1)$-ICRI hash function. In other words, an adversary who can attack the IMAP by mounting an impersonation attack with non-negligible probability can also win the ICRI game with non-negligible probability.

Consider the game illustrated in Fig. 3.12. If Eve wins this game with probability ϵ, then obviously we can translate the game into an attack against our IMAP with success probability ϵ. As a result, this game is named the "IMAP game". Here, Eve is simulating the adversary of the IMAP and is facing a challenger who is simulating Alice and Bob at the same time.

The first q rounds, analogous to the information gathering stage of an attack, consist of Eve sending messages M_i and the challenger responding with K_i. This part is simulating the first flow sent by Alice.

Eve is allowed to change the values sent by Alice and Bob sent over the insecure channel, that is K_i and R_i. Note that $h_i = H(M_i\|K_i\|R'_i)$. Hence, the values of K'_i

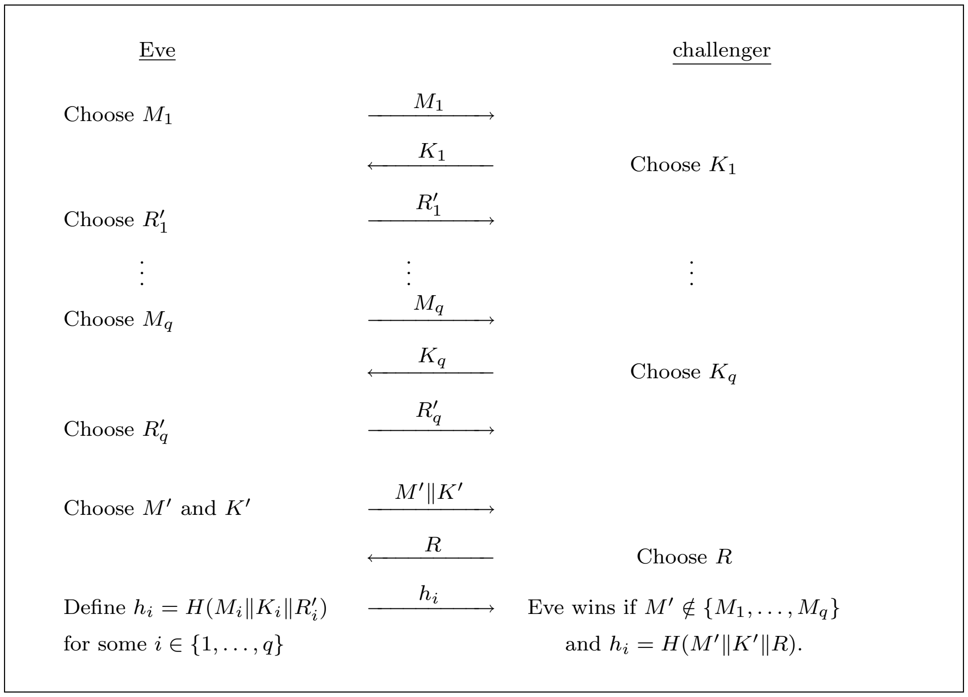


Figure 3.12: IMAP Game Against Impersonation Attacks

and R_i are redundant in the analysis of the impersonation attack.

In the last round of the game, corresponding to the deception phase, Eve sends $M'\|K'$, $M' \neq M_i$ for every $i \in \{1,\ldots,q\}$. After receiving a random value R from the challenger, she sends $h_i = H(M_i\|K_i\|R'_i)$, for some $i \in \{1,\ldots,q\}$. Eve wins the game if $h_i = H(M'\|K'\|R)$ for $M' \notin \{M_1,\ldots,M_q\}$.

The following theorem reduces the ICRI game to the IMAP game against impersonation attacks.

Theorem 2. *Let H be a $(T_{\text{off}}, \epsilon_1)$-ICRI hash function. Then, any adversary against the IMAP of Fig. 3.9 with offline complexity T_{off} who makes q message queries and mounts an impersonation attack, has a probability of success at most $q\epsilon_1$.*

Proof. Assuming that Eve wins the IMAP game of Fig. 3.12 with non-negligible probability, we can employ her in the ICRI game depicted in Fig. 3.6. In this reduction, Eve is playing against her IMAP game challenger and Oscar is playing against his ICRI game challenger. The result of the IMAP game, played by Eve, is going to be used in the ICRI game, played by Oscar. Oscar begins by choosing a random value $j \in \{1,\ldots,q\}$. Then, he lets Eve continue playing against the IMAP challenger, which is simulated by Oscar. Oscar does not interrupt the flows between Eve and her challenger except when $t = j$. For $t = j$, Oscar forwards M_j to the

ICRI challenger. Then, the challenger responds with K. Oscar sends $K = K_j$ to Eve. Oscar gets R' from Eve and sends it to the ICRI challenger.

At the deception stage, Eve sends M' and K'. Oscar sends M' to his challenger and receives R. He then sends R to Eve. Eve responds with a value h_i, $i \in \{1, \ldots, q\}$. Eve wins if $h_i = H(M' \| K' \| R)$. If $i = j$ and Eve wins, then Oscar wins the ICRI game, and Oscar loses otherwise.

If we assume that Eve can win the IMAP game with probability ϵ, then Oscar wins the ICRI game with probability ϵ/q. $\square$

When $q = 1$, adversaries with probability of success 2^{-k} clearly exist, and hence, the reduction is tight. For $q \neq 1$, the probability of success is $q2^{-k}$. This factor q appears as a consequence of considering strong adversaries who can request q messages to be sent by Alice. Some papers only consider $q = 1$ resulting in a weaker notion of security.[1] However, the approach of many other papers is similar to this thesis. For instance, Vaudenay [Vau05] assumed that $q \leq 2^{10}$ and the reduction is not tight. They also get the same probability of success, p/q.

Putting Lemma 1 and Theorem 2 together, we obtain the following corollary.

Corollary 2. *Let $\mathcal{X} = \{0,1\}^{\ell_1+\ell_2+\ell_3}$ be the set of all possible binary strings of size $\ell_1 + \ell_2 + \ell_3$ and H be a hash function chosen randomly from $\mathcal{F}^{\mathcal{X},\mathcal{Y}}$, where $|\mathcal{Y}| = 2^k$. Then, any adversary against the IMAP of Fig. 3.9, with offline complexity $T_{\text{off}} = 2^{t_{\text{off}}}$ who makes up to q message queries and mounts an impersonation attack, has a probability of success $p \leq q2^{-k}(1 + 2^{2t_{\text{off}}-\ell_2-\ell_3} + 2^{t_{\text{off}}-\ell_2})$.*

3.3.3.3 Substitution Attack

In the substitution attack, unlike the case of impersonation attack, Alice is actively involved and she would like to authenticate M to Bob. The adversary, on the other hand, wishes to authenticate M' to Bob along with the identity of Alice. There are two possible cases.

The first case is when Alice initiates a session and tries to authenticate M to Bob. Then, Eve substitutes M' for M. As a result, Bob receives M' and not M. The value of M' may be the result of a partial or total modification of M by Eve. After receiving R from Bob, Eve tries to find a suitable value R' which will make Bob accept after receiving h. Figure 3.13 is illustrating this scenario against our IMAP. This substitution attack is an attack of type ABAB, depicted in Figure 3.3.

[1] See [NSS06] for instance.

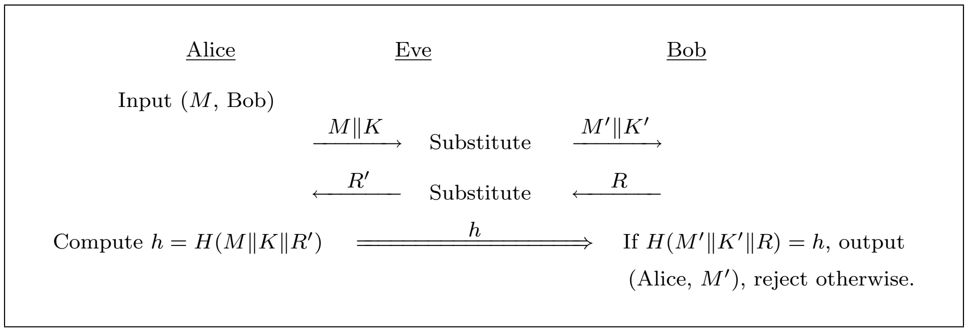

Alice
Eve
Bob
Input (M, Bob)
$M\|K$
Substitute
$M'\|K'$
R'
Substitute
R
Compute $h = H(M\|K\|R')$
h
If $H(M'\|K'\|R) = h$, output (Alice, M'), reject otherwise.

Figure 3.13: Substitution Attack of Type ABAB Against Our IMAP

The second case is when Eve initiates a flow with Bob while pretending to be Alice. Eve tries to authenticate M' to Bob. After receiving R, she does her computations to find a suitable M. Then, she will make Alice initiate a session with Bob with input M. Eve will use the authenticated flow of this session in her original session with Bob. This substitution attack is an attack of type BAAB, illustrated in Fig. 3.5.

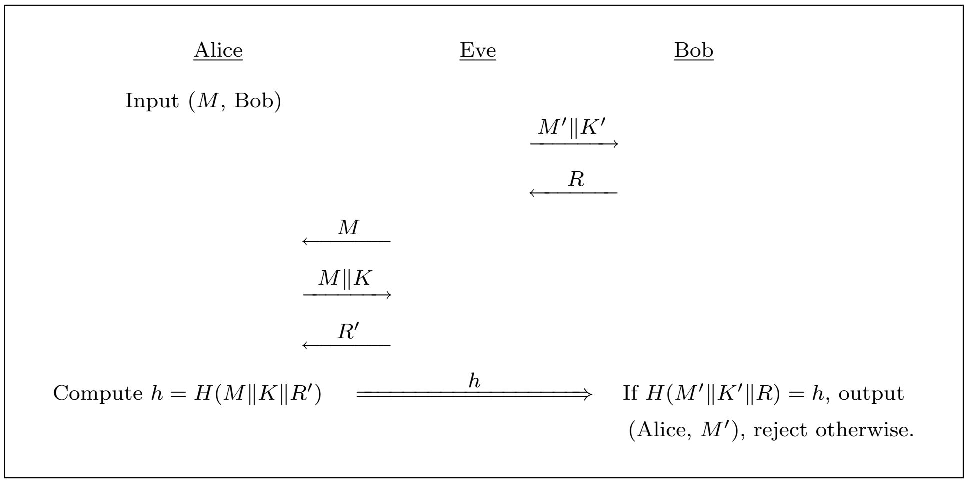

Alice
Eve
Bob
Input (M, Bob)
$M'\|K'$
R
M
$M\|K$
R'
Compute $h = H(M\|K\|R')$
h
If $H(M'\|K'\|R) = h$, output (Alice, M'), reject otherwise.

Figure 3.14: Substitution Attack of Type BAAB Against Our IMAP

The IMAP Game Against Substitution Attacks. Examining the substitution attack of type ABAB, illustrated in Fig. 3.13, we can write down the following as the order of the flows:

(1) Alice chooses M or gets it from Eve. Eve gets K from Alice.

(2) Eve sends M' and K' to Bob.

(3) Bob chooses a random value R and sends it to Eve.

(4) Eve chooses a random value R' and sends it to Alice.

(5) Alice computes $h = H(M\|K\|R')$, which is sent to Bob.

Note that a successful substitution attack of type ABAB directly translates into a successful player against the ICRII game. As a result, we get the following theorem.

Theorem 3. *Let H be a $(T_{\text{off}}, T_{\text{on}}, \epsilon_2)$-ICR*II *hash function. Then, any adversary against our IMAP with offline complexity T_{off} and online complexity T_{on}, who is mounting a substitution attack of type A has a probability of success $p = \epsilon_2$.*

Now we examine the substitution attack of type BAAB, illustrated in Fig. 3.14. The following is the order of the flows as they happen in this attack scenario:

(1) Eve sends M' and K' to Bob.

(2) Bob chooses a random value R and sends it to Eve.

(3) Eve provides Alice with M.

(4) Alice sends M and K to Eve.

(5) Eve chooses a random value R' and sends it to Alice.

(6) Alice computes $h = H(M\|K\|R')$, which is sent to Bob.

A successful substitution attack of type BAAB yields a successful player against the ICRIII game. Hence, the following theorem follows.

Theorem 4. *Let H be a $(T_{\text{off}}, T_{\text{on}}, \epsilon_2)$-ICR*III *hash function. Then, any adversary against our IMAP with offline complexity T_{off} and online complexity T_{on}, who is mounting a substitution attack of type BAAB has a probability of success $p = \epsilon_2$.*

Now combining Lemmas 2 and 3 with Theorems 3 and 4, we obtain the following corollary.

Corollary 3. *Let $\mathcal{X} = \{0,1\}^{\ell_1+\ell_2+\ell_3}$ be the set of all possible binary strings of size $\ell_1+\ell_2+\ell_3$ and H be a hash function chosen randomly from $\mathcal{F}^{\mathcal{X},\mathcal{Y}}$, where $|\mathcal{Y}| = 2^k$. Then, any adversary against our IMAP, with offline complexity $T_{\text{off}} = 2^{t_{\text{off}}}$ and online complexity $T_{\text{on}} = 2^{t_{\text{on}}}$, who is mounting a substitution attack, has a probability of success $p = 2^{-k}(1 + 2^{2t_{\text{off}}-\ell_2-\ell_3} + 2^{t_{\text{off}}-\ell_3} + 2^{t_{\text{on}}})$.*

3.3.3.4 Security of our IMAP

The adversary against our IMAP will either mount a substitution attack or an impersonation attack. Hence, the following theorem is a consequence of Corollary 2 and Corollary 3.

Theorem 5. *Let $\mathcal{X} = \{0,1\}^{\ell_1+\ell_2+\ell_3}$ and H be a hash function chosen randomly from $\mathcal{F}^{\mathcal{X},\mathcal{Y}}$, where $|\mathcal{Y}| = 2^k$. Then, any adversary against our IMAP, with offline complexity $T_{\text{off}} = 2^{t_{\text{off}}}$ and online complexity $T_{\text{on}} = 2^{t_{\text{on}}}$ who can make q message queries, has a probability of success*

$$p \leq 2^{-k} \max(q(1 + 2^{2t_{\text{off}}-\ell_2-\ell_3} + 2^{t_{\text{off}}-\ell_2}), 1 + 2^{2t_{\text{off}}-\ell_2-\ell_3} + 2^{t_{\text{off}}-\ell_3} + 2^{t_{\text{on}}}).$$

3.3.4 The Choice of Parameters and Hash Function

Theorem 5 says that an adversary attacking our proposed IMAP, using $2^{t_{\text{off}}}$ hash computations before the deception stage, $2^{t_{\text{on}}}$ hash computations during the deception stage, and q message queries, has a probability of success at most $2^{-k}\max(q(1+2^{2t_{\text{off}}-\ell_2-\ell_3} + 2^{t_{\text{off}}-\ell_2}), 1 + 2^{2t_{\text{off}}-\ell_2-\ell_3} + 2^{t_{\text{off}}-\ell_3} + 2^{t_{\text{on}}})$.

Let us first target typical[2] values for $q \leq 2^{10}$, $t_{\text{off}} \leq 70$, and $p \leq 2^{-20}$.

If we take $\ell_2, \ell_3 \geq 80$, then we can basically ignore the factors $(1 + 2^{2t_{\text{off}}-\ell_2-\ell_3})$ and $2^{t_{\text{off}}-\ell_3}$. We note that, since R and K are being sent over the insecure channel, this assumption does not have any impact on the analysis or usefulness of our protocol. We now can simplify the result of Theorem 5 to $p \leq 2^{-k}\max(q, 2^{t_{\text{on}}})$.

Since we want the overall success probability of the adversary to be less than or equal to 2^{-20}, we require that $\max(q, 2^{t_{\text{on}}}) \leq 2^{k-20}$.

Hence, letting $t_{\text{on}} = 10$ along with typical parameters $q \leq 2^{10}$, $t_{\text{off}} \leq 70$, and $p \leq 2^{-20}$, we get that $k \geq 30$. This is a distinct improvement over the previous works.

Vaudenay [Vau05] requires $k \geq 50$ while the same typical parameters are targeted. If we let $k = 50$, then we can tolerate much stronger adversaries, compared to other results [Vau05, MS07, PV06], having $t_{\text{on}} = 30$ and $q \leq 2^{30}$ and still get the same overall success probability of $p \leq 2^{-20}$. Note that, we can allow t_{off} to get bigger as well by just choosing $\ell_2 + \ell_3$ according to the size of t_{off}.

[2]See for instance [Vau05] and [PV06].

As a concrete suggestion, we would propose to use a standard hash function such as SHA-256, with the output truncated to k bits. This would certainly be practical. The issue remains as to whether it would be secure. We have proved that the protocol is secure in the Random Oracle Model, which is a standard approach in the design of cryptographic protocols. Furthermore, we also determined the exact properties of a hash function that are required for the security proof to hold. Of course we are not able to prove that these properties hold for any specific hash function. On the other hand, no one is able to prove at the present time that any specific hash function satisfies any desirable property (e.g., preimage-resistance). However, assuming a CR hash function is enough for ICRI, ICRII, and ICRIII properties to hold. As a result, if we are comfortable with using a concrete hash function in practice in place of a CR hash function, then we can also be comfortable with using that hash function when ICR hash functions are needed.

In practice, there needs to be a relation between the size of the messages M, ℓ_1, and the choice of t_{on}. Note that, in a substitution attack the adversary is making $2^{t_{\text{on}}}$ hash computations while Alice is waiting to get a value R from Bob. Generating a random value R does not take long. For our application, in particular, these devices are in close proximity and as a results the delay in the system should be low as well. This means that when Alice does not hear back from Bob, she suspects that some active adversary is trying to intervene.

3.4 An Unconditionally Secure IMAP

Naor, Segev and Smith [NSS06] proposed an unconditionally secure IMAP, with k rounds, using evaluation of polynomials over finite fields, for every integer k. To authenticate an n-bit message in k rounds, they require the length of the authenticated string to be about $2\log(1/\epsilon) + 2\log^{(k-1)} n + O(1)$, where ϵ is the probability of success of the adversary. The length of the authenticated string over the narrow-band channel is $2\log(1/\epsilon) + O(1)$ when $k = \log(n)$. Moreover, they proved that their protocol is essentially optimal by proving a lower bound of $2\log(1/\epsilon) - 6$ on the required length of the authenticated string. This result implies that the advantage of assuming computational security versus unconditional security is to reduce the amount of information that needs to be manually authenticated and not to reduce the number of rounds.

As with other protocols introduced in this chapter, we focus on unconditionally secure IMAPs with $k = 3$. We find sufficient conditions for an IMAP with three rounds to be unconditionally secure. On the one hand, our work is a special case of the work done by Naor et al. since we are only looking at protocols with three rounds. On the other hand, it is a generalization of their work because we do not limit ourselves to a particular polynomial construction over finite fields.

Let $\mathcal{M}$ be the set of all messages, let $\mathcal{K}$ be the set of all possible keys, and let $\mathcal{H}$ be a set of keyed hash functions of the form $h_y : \mathcal{M} \to \mathbb{F}_q$ for $y \in \mathcal{K}$. Figure 3.15 illustrates a generalization of the protocol proposed by Naor et al.

<u>Alice</u>		<u>Bob</u>
Input (M, Bob)		
Choose $x \in \mathbb{F}_q$ uniformly at random	$\xrightarrow{M, x}$	Receive M', x'
	$\xleftarrow{y}$	Choose $y \in \mathcal{K}$ uniformly at random
Receive y' and		
Compute $t = h_{y'}(M) + x$	$\xRightarrow{y', t}$	Output (Alice, M') if $y = y'$ and $t = h_y(M') + x'$, reject otherwise.

Figure 3.15: A Generalization of Naor-Segev-Smith IMAP

In order to analyze these attacks, we need the following standard definition for $\epsilon - \Delta U$ hash families.

Definition 5. *A hash family $\mathcal{H}$ is an $\epsilon - \Delta U$ hash family if for all choices of $M, M' \in \mathcal{M}$ and $x \in \mathbb{F}_q$,*

$$\Pr[h_y(M) - h_y(M') = x] \leq \epsilon,$$

where the probability is over all random choices of y.

For the protocol presented in Fig. 3.15, y is chosen uniformly at random from $\mathcal{K}$. Next, we examine each possible attack in more detail.

In a BAAB attack, Eve is required to set $y = y'$, otherwise she will be detected. She is successful if and only if

$$h_{y'}(M) + x = h_{y'}(M') + x'.$$

In other words, Eve succeeds if and only if $x = h_{y'}(M') + x' - h_{y'}(M)$. Since x is randomly chosen by Alice after y' is chosen by Bob, Eve succeeds with probability 2^{-q}.

In an AABB attack, on the other hand, Eve first receives M, x and has to guess the key y' ahead of time in order to set $y = y'$. Later, she can choose M' and x such that $h_y(M) + x = h_y(M') + x'$. The probability that Eve guesses the right key y' is $1/|\mathcal{K}|$.

Finally, in an ABAB attack, Eve receives M, x and fixes M', x' before y' is chosen by Bob. She is successful if and only if $h_{y'}(M) + x = h_{y'}(M') + x'$, or $h_{y'}(M) - h_{y'}(M') = x' - x$. Note that, $x' - x$ is fixed. Hence, if $\mathcal{H}$ is an $\epsilon - \Delta U$ hash family, then Eve succeeds with probability at most ϵ.

If we summarize the above three attacks, we see that Eve succeeds with probability $\max\{\epsilon, 2^{-q}, |\mathcal{K}|^{-1}\}$, and the size of the authenticator is $\log_2 |\mathcal{K}| + q$ bits. Note that there are 2^{-q} different possibilities for the value of x. Hence, we have $\epsilon \geq 2^{-q}$ and we can conclude that Eve's success probability is

$$\max\{\epsilon, |\mathcal{K}|^{-1}\}.$$

The protocol proposed by Naor et al. [NSS06] is a special case of our construction, where the $\epsilon - \Delta U$ hash family is constructed from a Reed-Solomon code. We now briefly describe their construction for a k-round protocol. For a message $m = m_1 \dots m_k \in (\mathbb{F}_q)^k$ and $x \in \mathbb{F}_q$, they define the polynomial $C_x(m) = \sum_{i=1}^{k} m_i x^i$. For distinct messages $m, m' \in (\mathbb{F}_q)^k$ and for any constants $c, c' \in \mathbb{F}_q$, we obtain

$$\Pr_{x \in_R \mathbb{F}_q} [C_x(m) + c = C_x(m') + c'] \leq \frac{k}{q}.$$

The polynomial $C(.)$ is used as a hash function k times, once in each round. Each application of the hash function results in a shorter message. After k applications, the output message is manually authenticated.

Chapter 4

Message Recognition Protocols

Contents

We look at the problem of message and entity recognition by reviewing the definitions and the security model described in the literature. In Section 4.1, we prove that there is a one-to-one correspondence between non-interactive message recognition protocols and digital signature schemes with message recovery. Further, in Section 4.2, we examine previous recognition protocols proposed in the literature [ABC$^+$98, HWGW05, LZWW05, Mit03, WW03], and conclude that the protocol of Lucks et al. [LZWW05] is more suitable compared to other proposals in the literature. Hence, we look at this protocol in more detail and we will refer to it as the Lucks protocol for short. In Section 4.3, we suggest a variant to overcome a certain shortcoming. In particular, the Lucks protocol is not equipped with a practical resynchronization process and can fail to resume in case of communication failure or adversarial disruption. We propose [MS08c] a variant of this protocol which is equipped with a resynchronization technique that allows users to resynchronize whenever they wish or when they suspect an intrusion.

On the other hand, it is also of interest to remedy this shortcoming without requiring a separate synchronization procedure. In Section 4.4, we propose a new message recognition protocol [GMS08], which is based on the original protocol by Lucks et al., and incorporates the resynchronization technique within the protocol itself. That is, without having to provide a separate resynchronization procedure, we overcome the recoverability problem of the protocol. Moreover, we analyze all possible attacks against our protocol and prove that they can succeed with negligible probability. We further prove the security of the protocol in the model described in Section 1.3, and its ability to self-recover once the disruption has stopped.

Finally, in Section 4.5, we propose a message recognition protocol without the use of hash chains [MS08b] which is suitable for ad hoc pervasive networks. Hence, we no longer require the devices to save values of a hash chain in their memories. This relaxes the memory requirements. Moreover, we do not need to upperbound the total number of times the protocol can be executed which implies a desired flexibility in this regard. Furthermore, our protocol is secure without having to consider families of assumptions that depend on the number of times the protocol is executed. Hence, the security does not weaken as the protocol is executed over time. Last but not least, we provide a practical procedure for resynchronization in case of any adversarial disruption or communication failure.

4.1 Non-interactive MRPs

In this section, we prove that there is a one-to-one correspondence between non-interactive message recognition protocols and digital signature schemes with message recovery.

4.1.1 A General Non-Interactive MRP

A general non-interactive message recognition protocol, where all flows are going from Alice to Bob, consists of two flows. The first flow is the initialization step which happens only once. The second flow, occurring over the insecure channel, is sent once for each message to be authenticated. As a result, the message and its commitment (and possibly some other information) are all being transmitted in one flow. Hence, one should not reveal keys, such as a_i in the hash chain, in these protocols (otherwise, the adversary having seen the revealed key will stop this single flow and commit to a message of her own using this key). This implies that there is no point in using hash chains or any form of chains in the non-interactive setting since the chains can only be useful when you actually reveal them.

Figure 4.1 depicts a general non-interactive message recognition protocol. On input (1^k), where k is a security parameter, the function f outputs a pair of keys (a, A). In order to make impersonation impossible, it should be hard to find a given the value of A. The protocol is described in terms of two functions, denoted by *compose* and *decompose*. The function *compose* can be a randomized algorithm. Note that any non-interactive protocol can be put in this form. It is required that $\mathit{decompose}(c', A) = \perp$ with high probability if $c' \neq \mathit{compose}(M, a)$ for some message M and a valid pair of keys (a, A). Moreover, it is required that $\mathit{decompose}(c, A) = M$ when $c = \mathit{compose}(M, a)$.

<u>Alice</u>	<u>Eve</u>	<u>Bob</u>
compute $f(1^k) = (a, A)$	$\xRightarrow{A}$	Receive A.
<u>Alice</u>	<u>Eve</u>	<u>Bob</u>
Input (M, Bob)		
$\mathit{compose}(M, a) = c$	$\xrightarrow{c}$	Receive c'
		Compute $d = \mathit{decompose}(c', A)$ If $d = M'$, a valid message, output (M', Alice), otherwise, reject.

Figure 4.1: A Non-interactive Message Recognition Protocol

4.1.2 Digital Signature Schemes with Message Recovery

Digital signature schemes with message recovery (DSSMR), Fig 4.2, are often formulated by three algorithms: key generation, sign and verify. The key generation algorithm $\mathcal{G}$ randomly produces a pair of public and private keys (PK, SK) for each signer. The signer uses SK to sign and PK is used by others to verify signatures. On input message M and a secret key SK, the signing algorithm $\mathcal{S}$, which may be randomized, outputs a signature s. On input public key PK and a signature s, the signature verifying algorithm, $\mathcal{V}$, either outputs M', a valid message, or it rejects s.

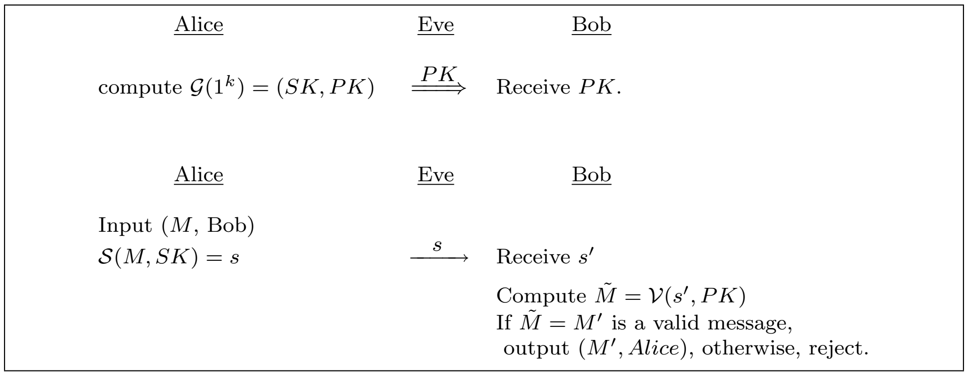

Alice
Eve
Bob
compute $\mathcal{G}(1^k) = (SK, PK)$ $\xRightarrow{PK}$ Receive PK.
Alice
Eve
Bob
Input (M, Bob)
$\mathcal{S}(M, SK) = s$ $\xrightarrow{s}$ Receive s'
Compute $\tilde{M} = \mathcal{V}(s', PK)$
If $\tilde{M} = M'$ is a valid message,
output $(M', Alice)$, otherwise, reject.

Figure 4.2: A Digital Signature Scheme with Message Recovery

A signature s of M that is honestly computed, using the secret key SK, should be accepted by the verifying algorithm using the associated public key PK. In other words, for all M, PK, and SK, it holds that $\mathcal{V}(PK, \mathcal{S}(M, SK)) = M$. Further, it should be difficult for any polynomially bounded adversary, to forge valid signature(s) knowing only the public key PK, and the three algorithms.

4.1.3 Equivalence of Non-interactive MRPs and DSSMRs

Given the aforementioned definitions and properties, we obtain the following obvious result on the equivalence of digital signature schemes with message recovery and non-interactive message recognition protocols.

Theorem 6. *Given functions f, compose, and decompose, any non-interactive message recognition protocol can be transformed to a digital signature scheme with message recovery. Conversely, any digital signature scheme with message recovery, with functions $\mathcal{G}, \mathcal{S}$, and $\mathcal{V}$, can be transformed to a non-interactive message recognition protocol.*

By letting $SK := a$, $\mathcal{G} := f$, $\mathcal{S} :=$ *compose*, and $\mathcal{V} :=$ *decompose*, the forward statement clearly follows. Similarly, the converse statement follows by letting $a := SK$, $f := \mathcal{G}$, *compose* $:= \mathcal{S}$, and *decompose* $:= \mathcal{V}$.

Note that it was previously known that any signature scheme could be used to construct a non-interactive recognition protocol, which is a more general case of the converse statement of the theorem. However, the forward result is new.

4.2 Previous MRPs

In this section, we review the existing protocols in the literature which provide entity or message recognition. The usability of each protocol is discussed in the context of networks with devices having low computation power and low communication bandwidth.

The 'Guy Fawkes protocol' was proposed by Anderson et al. [ABC+98]. There are two variants of this protocol suggested and a one-way hash function is deployed in both variants. In the first variant, random codewords, X_i, are chosen in each session and are refreshed each time a message, M_i, is authenticated. Alice commits to the message and the codewords and then publishes the commitment in a public directory which provides time-stamping services. Later, she reveals the committed values to prove that she is the same party who was involved in previous sessions. However, assuming the existence of a trusted party which provides time-stamping services is not realistic in most ad hoc network scenarios. The second variant does not require any interaction with a time-stamping provider and instead requires interaction of the authenticating party with the verifying party. The initialization phase of this protocol does not assume any authenticated channel; however, it requires digital signatures for authenticating the first blocks and codewords. This may not be suitable in ad hoc networks and, in particular, in low-power environments. Moreover, for a message to be authenticated in session i, users need to commit to it in the previous session. In the context of message recognition, this means that users are engaged in two sessions of this protocol to authenticate a single message, which may not be desirable.

An entity recognition protocol, known as 'Remote User Authentication Protocol', was introduced by Mitchell [Mit03]. In this protocol, a message authentication code (MAC) is used to prove that a user is the same entity involved in previous sessions. The protocol can be adapted to perform message recognition as well; however, this is not discussed in the paper. The setup phase of this protocol requires that t MAC values be sent over the authenticated channel. This may be costly since authenticated channels are usually of low bandwidth. Further, the "cut-and-choose" procedure in each round involves in sending $2t$ MAC values and r secret keys. In order for the protocol to be secure, it is suggested that $t \geq 35$ and $r \approx t/2$. Hence, the amount of computation and communication here is large compared to other protocols that are providing entity or message recognition and it may not be suitable for settings with low power devices.

Weimerskirch et al. introduced a protocol called 'Zero Common-Knowledge' (ZCK) protocol [WW03]. They use MACs and hash chains of the form $a_i = h(a_{i-1})$ and $b_i = h(b_{i-1})$, $i = 1, \ldots, n$, as keys for the MACs computed by Alice and Bob, respectively. Here, n is fixed at the beginning and h is a one-way hash function. Hammell et al. implemented the ZCK protocol [HWGW05]. The provided measurements and observations from this implementation gave a proof-of-concept. Low computational power, low code space, low communication bandwidth, low energy resources, are among the main requirements of a recognition protocol designed for an ad hoc pervasive network setting. The measurements resulted from this implementation proved that the ZCK protocol exhibits the aforementioned requirements.

Note that Hammell et al. [HWGW05] investigate the practicality of the ZCK protocol but do not investigate its security properties. That is, Hammell et al. rely on the security proof presented by Weimerskirch et al. [WW03]. However, Lucks et al. found a flaw in the security proof of this protocol and presented an attack against the ZCK protocol. Furthermore, they proposed a modification to fix the flaw.

4.2.1 The Lucks Protocol

As noted above, Lucks et al. [LZWW05] found an attack against the ZCK protocol and pointed out the flaw in the security proof of this protocol. Further, using the same idea of using values in a hash chain as keys for MACs, they proposed a message recognition protocol which is a modification of the original ZCK protocol.

They consider a cryptographic hash function $h : \{0,1\}^s \rightarrow \{0,1\}^s$ as a one-way hash function, and a message authentication code MAC : $\{0,1\}^s \times \{0,1\}^* \rightarrow \{0,1\}^c$. Typical values are suggested to be $s \geq 80$ and $c \geq 30$. Further, n is fixed to be the maximum number of messages to be authenticated. In other words, the maximum number of sessions is fixed to be n. Alice and Bob randomly choose a_0 and b_0, respectively. Then, they respectively form $a_i = h(a_{i-1})$ and $b_i = h(b_{i-1})$, $i = 1, \ldots, n$. In the initialization phase, Eve is assumed to be passive. Hence, we can denote this channel, in accordance with our notation, by $\Rightarrow$. Alice and Bob will exchange a_n and b_n over the authenticated channel during this phase.

Alice		Bob
Choose a random a_0 and compute $a_i = H(a_{i-1})$ for $i = 1, \ldots, n$	$\xRightarrow{a_n}$	Choose a random b_0 and compute $b_i = H(b_{i-1})$ for $i = 1, \ldots, n$.
	$\xLeftarrow{b_n}$	

Figure 4.3: Initialization Phase of the Lucks Recognition Protocol

After the initialization phase, there are n sessions denoted by $n-1, \ldots, 0$, starting from session $n-1$ and moving down to lower values one at a time. In session i, Alice authenticates the message m_i using a_i as the key for the MAC. Once Bob authenticates himself to Alice by revealing b_i, Alice reveals a_i and allows Bob to verify and accept this new key and the authenticity of the message m_i. When a key k is accepted, it is denoted by accept-key(k). Moreover, commit-message(m, i) indicates that Alice commits to a message m in session i, and accept-message(m, i) indicates that Bob accepts m as authentic and fresh in session i. After a successful session of the protocol, Alice and Bob will "move down" in the hash chain, using a_{i-1} and b_{i-1} for session $i-1$.

Alice's internal state in the Lucks protocol is as follows:

- i, the session counter
- b_{i+1}, the most recently accepted value of Bob's hash chain (hence accept-key(b_{i+1}) has occurred already)
- a one-bit flag, to distinguish the program states **A0** and **A1**.

Similarly, Bob's internal state is:

- i, the session counter
- a_{i+1}, the most recently accepted value of Alice's hash chain (hence accept-key(a_{i+1}) has occurred already)
- a one-bit flag, to distinguish the program states **B0** and **B1**.

Session i of the Lucks protocol:

A0 (Alice's initial program state) Obtain m_i (possibly from Eve), then

```
        Commit-message($m_i, i$).
        Compute $d_i = \mathrm{MAC}_{a_i}(m_i)$.
        Send $(d_i, m_i)$; goto **A1**.

A1  Wait for a message $b'$ (supposedly from Bob), then
        If $H(b') = b_{i+1}$ then
          Let $b_i := b'$, accept-key($b_i$) and send $a_i$. Let $i := i - 1$ and goto **A0**
        else goto **A1**.

B0  (Bob's initial program state) Wait for a message $(d_i, m_i)$, then send $b_i$ and
    goto **B1**.

B1  Wait for a message $a'$ (supposedly from Alice), then
        If $H(a') = a_{i+1}$ then
          Let $a_i := a'$ and accept-key($a_i$).
          If $\mathrm{MAC}_{a'}(m_i) = d_i$ then
            Accept $m_i$ as authentic in session $i$
          (else do not accept any message for session $i$).
          Let $i := i - 1$ and goto **B0**
        else goto **B1**.
```

Figure 4.4 depicts the Lucks protocol. We analyze this protocol in more detail and point out its shortcomings in case of adversarial disruption or communication failure. Further, we propose a new variant of the recognition protocol of Lucks et al. which incorporates a resynchronization technique allowing a full recoverability of the protocol.

Lucks et al. present their protocol in an extended abstract [LZWW05], and prove its security in the full version of the paper [LZWW07]. An updated version of this proof is also published in [LZWW08]. The protocol is proved to be secure given that the the following properties hold for the hash function H and the message authentication code MAC.

Definition 6. *Let secret $y_0, y_1, \ldots, y_i$ and known y_{i+1} be chosen such that $y_{i+1} = H(y_i)$, $y_i = H(y_{i-1}), \ldots, y_1 = H(y_0)$. A hash function H is referred to as a* **depth-i preimage resistant (i-PR)** *hash function when it is infeasible to find y' such that $y_{i+1} = H(y')$.*

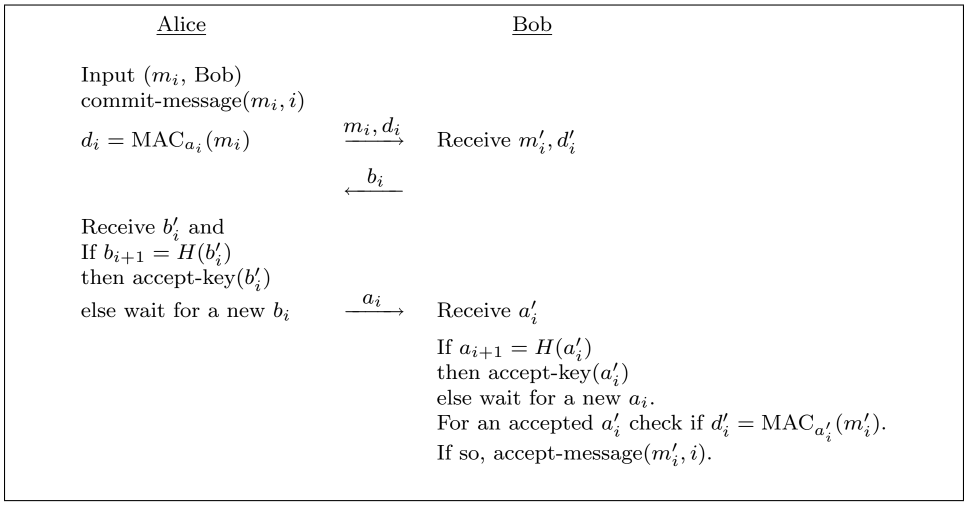


Figure 4.4: The Lucks Entity and Message Recognition Protocol

Definition 7. *Let secret $y_0, y_1, \ldots, y_{i-1}$ and known y_i, y_{i+1} be chosen such that $y_{i+1} = H(y_i)$, $y_i = H(y_{i-1}), \ldots,$ $y_1 = H(y_0)$. A hash function H is* **depth-i second preimage resistant (i-SPR)** *when it is infeasible to find y', $y' \neq y_i$, such that $y_{i+1} = H(y')$.*

Definition 8. *Let secret $y_0, y_1, \ldots, y_i$ and known y_{i+1} be chosen such that $y_{i+1} = H(y_i)$, $y_i = H(y_{i-1}), \ldots,$ $y_1 = H(y_0)$. A message authentication code MAC is* **depth-i existentially unforgeable** *if it is infeasible to mount an existential forgery against MAC_{y_i} in an adaptive chosen message attack scenario.*

4.2.1.1 Unrecoverability Problem of the Luck Protocol

There is a "small time-frame" associated with each session i. In particular, a message m_i is *fresh* if it is sent within the associated time-frame of session i. It is assumed that during each time-frame, Alice commits to only one message and Bob accepts at most one message. As a result, the time-frame should be known to both Alice and Bob. However, the value i, which could indicate the appropriate time-frame, is contained in the internal states of Alice and Bob. Note that i is not being transmitted during the protocol execution and it is implicit that Alice's and Bob's internal states agree on this value. This may be problematic in different ways. First, how will Alice and Bob remain synchronized during the different time-frames? Assuming a secure synchronized clock is a quick fix to this problem. However, assuming availability of such a service may not be practical for most ad hoc network scenarios. In particular, Lucks et al. assume that no securely synchronized clock is available. Hence, the process of synchronization is highly dependent on the schedule of received and sent messages, that is, on the dynamics of the communication in the network. This gives rise to the second problem: in case of communication failure or adversarial disruption, this protocol is not equipped with a practical resynchronization process.

We observe that although the Lucks protocol is provably secure, it nonetheless falls short in case of the following adversarial disruption. Eve can easily manipulate one party to move forward to the next session, while the other party is still in the previous session. In such a case, a party could get trapped in a state and never be able to finish execution of a session; as a result, he or she remains stuck in that state forever. It is also mentioned in [LZWW08] that Eve is able to stretch a session at her will.

Figure 4.5 illustrates a situation where Bob is trapped by Eve in program state **B1**. The condition in program state **B1** fails since $a_{i+1} \neq H(a'_i)$. This will cause Bob to stay in **B1** waiting for a new a_i. Now even if Alice sends him a legitimate message m_i, he will ignore it. Although this looks like a denial of service attack, it is much stronger than that. Eve can go away and yet Alice and Bob are still unable to communicate because Bob is trapped. The details of the disruption are as follows.

Eve sends m'_i and d'_i to Bob and he will automatically decrement his index to i while Alice does not. Eve chooses a'_i such that $a_{i+1} \neq H(a'_i)$, which will make Bob wait for a new a_i. While he is waiting for a new a_i, he will not accept a message of the form (m_j, d_j), for any j. Hence, even if Alice sends him a legitimate message, he will ignore it. As a result, he is "trapped" in state **B1**.

Lucks et al. [LZWW05] suggest that Bob sends b_i again after he has waited for too long to receive the correct a_i. However, when Alice has not initiated the session and is not anticipating b_i, it is not clear what she is supposed to do. Hence, this will not help the protocol recover in case of this particular disruption.

<u>Eve</u>		<u>Bob</u>
Choose random m'_i and d'_i.	$\xrightarrow{m'_i, d'_i}$	Move to the next time-frame upon reception of the new message.
	$\xleftarrow{b_i}$	
Choose a'_i such that $a_{i+1} \neq H(a'_i)$.	$\xrightarrow{a'_i}$	Since $a_{i+1} \neq H(a'_i)$, wait for a new a_i.

Figure 4.5: Eve "trapping" Bob in state **B1**

Eve can play the same trick with Alice and trap her in program state **A1** for an indeterminate period of time; Figure 4.6 illustrates this situation.

Alice		Eve
Input (m_i, Bob).		
commit-message(m_i, i).		
Compute $d_i = \mathrm{MAC}_{a_i}(m_i)$.	$\xrightarrow{m_i, d_i}$	
Since $b_{i+1} \neq H(b'_i)$, wait for a new b_i.	$\xleftarrow{b'_i}$	Choose b'_i such that $b_{i+1} \neq H(b'_i)$.

Figure 4.6: Eve "trapping" Alice in state **A1**

Once again, we note that this inability to recover is a problem since the adversary does not need to continue her active involvement. She can leave the network and yet Alice and Bob will no longer be able to have successful communication.

There should be a mechanism to help Alice and Bob resynchronize after having waited for a sufficiently long period of time for a new a_i or b_i. Otherwise, the protocol cannot be resumed and recoverability is lost. One way to perform this resynchronization is to utilize the authenticated channel occasionally. The advantage of this solution is that it is very simple. However, the authenticated channel is expensive and it may not be practical to assume that it is accessible after the initialization phase. For instance, the devices may be widely dispersed, and it may not be possible to collect them again to perform this kind of resynchronization. Furthermore, periodic employment of the resynchronization process, according to a predefined schedule, will not be based on the dynamics of the network. For instance, some devices may be more active than others or there may be more noise present in some parts of the network compared to other parts of the network. Indeed, there is more disruption caused by noise or communication failure in busier parts of the network. Hence, resynchronization among some users may be necessary more often than others. As a result, it is desirable to execute the resynchronization process when it is needed according to the state of the network. We propose the following protocol to overcome these shortcomings. We use the same hash function, H, used by Lucks et al. and write H^j, $j \geq 1$, to denote the case when the hash function H is applied j times iteratively.

4.3 An Improved MRP with Resynchronization Process

The internal state of Alice and Bob includes:

- i_A and i_B, counters pointing to the position of Alice and Bob in their respective hash chains,

- $i_{acceptA}$, a counter kept by Alice which is the smallest index such that Alice has accepted the key $b_{i_{acceptA}}$ in session $i_{acceptA}$. Similarly, $i_{acceptB}$, a counter kept by Bob which is the smallest index such that Bob has accepted the key $a_{i_{acceptB}}$ in session $i_{acceptB}$.

Alice executes the protocol as follows:

- Let $i := i_A$ and $j_A := i_{acceptA} - i_A$;
- Wait for m_i (possibly from Eve), then
- commit-message(m_i, i);
- compute $d_i = \mathrm{MAC}_{a_i}(i \| m_i)$;
- send $(i \| m_i, d_i)$;
- wait for a message b'_i (supposedly from Bob), then
  if $H^{j_A}(b'_i) = b_{i_{acceptA}}$, (key verification step)
  then $b_i := b'_i$; accept-key(b_i); send a_i; set $i_{acceptA} := i$ and $i_A = i - 1$;
  else initiate the resynchronization process.

Bob executes the protocol as follows:

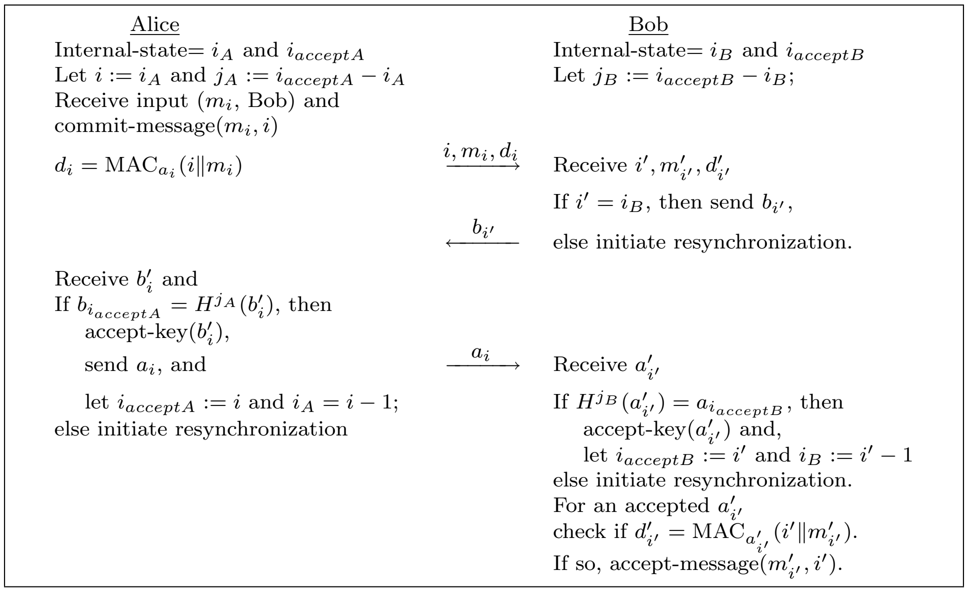
- Let $j_B := i_{acceptB} - i_B$;
- Wait for a message $(i' \| m'_{i'}, d_{i'})$.
- If $i' = i_B$, then send $b_{i'}$, else initiate the resynchronization process.
- Wait for a message $a'_{i'}$ (supposedly from Alice), then
  if $H^{j_B}(a'_{i'}) = a_{i_{acceptB}}$, (key verification step)
  then $a_{i'} := a'_{i'}$; accept-key$(a_{i'})$; set $i_{acceptB} := i'$ and $i_B := i' - 1$
  if $\mathrm{MAC}_{a_{i'}}(i' \| m'_{i'}) = d_{i'}$
  then accept $m'_{i'}$ as authentic in session i';

  else initiate the resynchronization process.

Figure 4.7 illustrates this protocol. Let us first highlight the differences between this protocol and the protocol of Lucks et al. In the internal states of Alice and Bob, the session counter i is replaced by i_A and i_B to incorporate adversarial ability to manipulate a party to decrement the session counter, as was discussed previously, and consequently change its position in the hash chain. For the same reason, $i+1$ is changed to $i_{acceptA}$ and $i_{acceptB}$ as the smallest index such that a key has been accepted by Alice or Bob, respectively. Moreover, a_{i+1} and b_{i+1} are replaced by $a_{i_{acceptB}}$ and $b_{i_{acceptA}}$ as the accepted keys. Further, parameters j_A and j_B are introduced to deal with the case where $i_{acceptA} > i_A + 1$ or $i_{acceptB} > i_B + 1$, respectively, due to an adversary's intrusions. A related modification refers to the *key verification* step, where the users may need to apply the hash function H more than once. In the protocol of Lucks et al., the session counter is not being transmitted or committed to by either party. However, we require that Alice commits to i_A and transmits it in the first flow. This allows Bob to easily detect any possible manipulations of the session counter by Eve. Furthermore, we provide a resynchronization process, allowing Alice and Bob to initiate the resynchronization process when they do not receive the correct keys. Hence, the adversary can no longer "trap" them in states $A1$ or $B1$, as was explained previously.

Surely, it holds that $i_A = i_B$ when the adversary has been passive since the initialization. Moreover, in the case where all flows are safely relayed from the initialization, Alice and Bob will accept every single key from the other party and move forward in the hash chain together. Hence, in the ith session, $i_A = i_B = i$ and $i_{acceptA} = i_{acceptB} = i + 1$. In particular, $j_A = i_{acceptA} - i_A = 1$ and $j_B = i_{acceptB} - i_B = 1$. However, once the adversary begins sending messages to Alice and Bob, she is capable of manipulating either party to decrement their session counter in a bogus session. Hence, Alice and Bob will need to resynchronize to agree on a mutual position in their respective hash chains, which may result in $j_A \neq 1$ or $j_B \neq 1$.

Note that this variant is instructing Alice and Bob to initiate resynchronization whenever a mismatch occurs. Hence, once the adversary initiates a bogus session, she can no longer continue another bogus session undetected. That is, she can make Alice and Bob decrement their session counters at most once. Hence, we have $|i_A - i_B| \leq 1$. In case of an active intrusion, the participants are not supposed to accept the key Eve sends them and, as a result, the values of $i_{acceptA}$ and $i_{acceptB}$ are not going to be updated. Consequently, we obtain $j_A = i_{acceptA} - i_A = 2$ or $j_B = i_{acceptB} - i_B = 2$.



Alice

Internal-state= i_A and $i_{acceptA}$
Let $i := i_A$ and $j_A := i_{acceptA} - i_A$
Receive input (m_i, Bob) and
commit-message(m_i, i)
$d_i = \mathrm{MAC}_{a_i}(i \| m_i)$

Bob

Internal-state= i_B and $i_{acceptB}$
Let $j_B := i_{acceptB} - i_B$;

$\xrightarrow{i, m_i, d_i}$

Receive $i', m'_{i'}, d'_{i'}$
If $i' = i_B$, then send $b_{i'}$,
else initiate resynchronization.

$\xleftarrow{b_{i'}}$

Receive b'_i and
If $b_{i_{acceptA}} = H^{j_A}(b'_i)$, then
accept-key(b'_i),
send a_i, and
let $i_{acceptA} := i$ and $i_A = i - 1$;
else initiate resynchronization

$\xrightarrow{a_i}$

Receive $a'_{i'}$
If $H^{j_B}(a'_{i'}) = a_{i_{acceptB}}$, then
accept-key($a'_{i'}$) and,
let $i_{acceptB} := i'$ and $i_B := i' - 1$
else initiate resynchronization.
For an accepted $a'_{i'}$
check if $d'_{i'} = \mathrm{MAC}_{a'_{i'}}(i' \| m'_{i'})$.
If so, accept-message($m'_{i'}, i'$).

Figure 4.7: Our Proposed Variant of the Lucks Protocol

In this protocol, the session counter is being transmitted in the first flow. Moreover, Alice commits to this value as part of the message, so the adversary cannot arbitrarily change it without being detected. This implies that the security proof of the Lucks et al. protocol will apply to this new variant as well. Furthermore, once either user realizes that Eve could have manipulated the values, they can initiate a resynchronization process. This process allows them to agree on a session counter $i_A = i_B$, which indicates the corresponding position of each user in their respective hash chains.

4.3.1 Resynchronization Process

At some point during the execution of the protocol, either Alice or Bob realizes the need for resynchronizing with the other party. This may be due to a mismatch caused by adversarial efforts or just due to some communication failure or noise. In the resynchronization process, Alice computes

$$I_A := \min\{i : \text{ Alice has revealed } a_i\} - 1$$

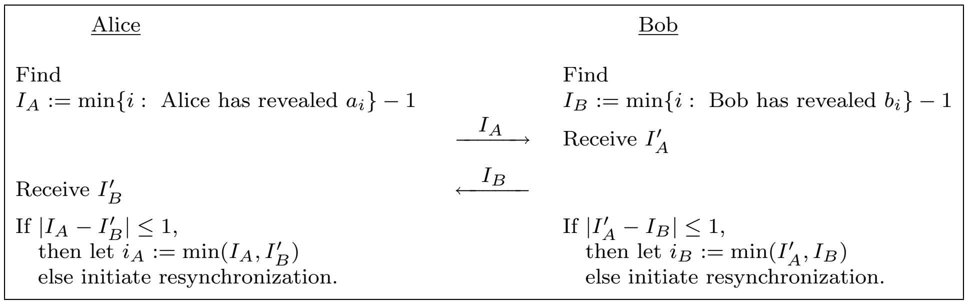
and, similarly,

$$I_B := \min\{i : \text{ Bob has revealed } b_i\} - 1$$

is computed by Bob. Then, they exchange I_A and I_B over the insecure channel. Note that, Eve can change these values, say to I'_A and I'_B, since they are being sent over the insecure channel.

Recall that we are instructing Alice and Bob to resynchronize whenever they notice a mismatch. This implies that the adversary, or some noise in the channel, can make them increment their session counters at most once before they try to resynchronize again. Hence, we have $|I_A - I_B| \leq 1$. This fact alone does not enable Alice and Bob to detect Eve's manipulation with I_A and I_B. However, it makes it impossible for Eve to choose values for I'_A and I'_B which are smaller than $I_A - 1$ and $I_B - 1$, respectively. We emphasize that this feature is important here. In the absence of such a feature, Eve can choose I'_A and I'_B to be very small and exhaust the hash chains too quickly. That would constitute a strong denial of service attack that can be prevented as follows.

Alice checks to make sure $|I_A - I'_B| \leq 1$ and Bob checks to see if $|I'_A - I_B| \leq 1$ holds. If either of these do not hold, it means that the adversary is attempting to intrude while they are trying to resynchronize. If $|I_A - I'_B| \leq 1$ and $|I'_A - I_B| \leq 1$ hold, then Alice and Bob will let $i_A := \min(I_A, I'_B)$ and $i_B := \min(I'_A, I_B)$, respectively. Figure 4.8 depicts the resynchronization process.



Alice		Bob
Find $I_A := \min\{i : \text{ Alice has revealed } a_i\} - 1$		Find $I_B := \min\{i : \text{ Bob has revealed } b_i\} - 1$
	$\xrightarrow{I_A}$	Receive I'_A
Receive I'_B	$\xleftarrow{I_B}$	
If $\|I_A - I'_B\| \leq 1$, then let $i_A := \min(I_A, I'_B)$ else initiate resynchronization.		If $\|I'_A - I_B\| \leq 1$, then let $i_B := \min(I'_A, I_B)$ else initiate resynchronization.

Figure 4.8: Resynchronization Process for the Proposed Protocol

Note that an active adversary can always disrupt the synchronization. When one party realizes this, he or she will call for a resynchronization again. If the adversary is passive in the resynchronization stage, then $I_A = I'_A$ and $I_B = I'_B$. As a result, $i_A = i_B$ and synchronization is achieved.

However, we will show that intrusions of an active adversary during the resynchronization stage, resulting in $i_A \neq i_B$, is going to be detected by either Alice or Bob. In case of intrusions where $|I_A - I'_B| > 1$ or $|I'_A - I_B| > 1$, the adversary is detected right away as discussed above. The rest of the intrusions are detected in the first session of the protocol immediately after the resynchronization, depicted in Fig 4.9. We show that the adversary is detected unless she has found unrevealed preimages of particular values in the hash chain. Note that i_A and i_B cannot differ very much, due to the conditions $|I_A - I'_B| \leq 1$ and $|I'_A - I_B| \leq 1$. However, we can prove the same statement even if the difference between i_A and i_B is not bounded.

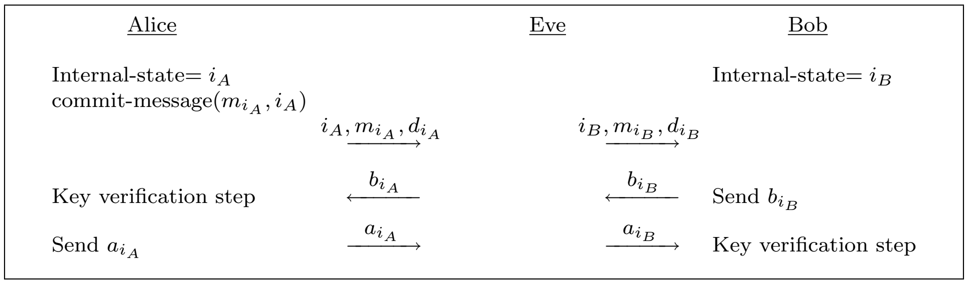


Figure 4.9: The First Execution after the Resynchronization

In order for Eve not to be detected by Bob in the key verification step, she must replace a_{i_A} with a_{i_B}. Otherwise, Bob will not accept the key and he will initiate resynchronization regardless of the values of m_{i_B} and d_{i_B}. Similarly, she has to replace b_{i_B} with b_{i_A}, otherwise, she will be detected by Alice. Now, assume that $i_A < i_B$ after the resynchronization. Finding a correct value for b_{i_A} means that Eve has found a nonempty chain of preimages $(b_{i_B}, b_{i_B-1}, \ldots, b_{i_A+1})$. Similarly, if $i_A > i_B$ and the adversary goes undetected, she has found a chain of preimages $(a_{i_A}, a_{i_A-1}, \ldots, a_{i_B+1})$. Hence, as long as finding preimages of H is a hard task, the adversary will be detected with high probability. As a result, we obtain the following theorem.

Theorem 7. *Let H be a depth-i second preimage resistant and depth-i preimage resistant hash function in the protocol of Fig 4.7. Consider a polynomially bounded adversary who changes the values of I_A or I_B in the resynchronization process of Fig 4.8, resulting in $i_A \neq i_B$. An undetected such intrusion can only occur with a negligible probability.*

4.4 An Improved MRP with Self-Recoverability

We describe the details of our proposed recognition protocol [GMS08] in this section, while the security and recoverability analyses are postponed to Section 4.4.1. Although this protocol is based on the original protocol proposed by Lucks et al., the logic of the instructions of Alice and Bob has changed considerably. Moreover, the information exchanged between Alice and Bob has changed as well.

Note that each pair of users can execute this protocol. However, there must be a different pair of hash chains for each pair of communicating users. It is implicitly assumed that Alice and Bob are the communicating parties in the rest of the paper.

The initialization phase and the setup of the hash chains are exactly as in the Lucks protocol. The internal state of Alice includes (along with each variable's initial value):

- $i_A := n - 1$: the position of Alice in her chain.
- $i_{acceptA} := n$: the last index of Bob's chain that was accepted by Alice.
- $b_A := b_n$: the last value of Bob's chain that was accepted by Alice.
- $M := Null$: the input message to be authenticated in the current session.
- a one-bit flag, to distinguish the program states **A0** and **A1**.

Similarly, Bob's internal state is as follows:

- $i_B := n - 1$: the position of Bob in his chain.
- $i_{acceptB} := n$: the last index of Alice's chain that was accepted by Bob.
- $a_B := a_n$: the last value of Alice's chain that was accepted by Bob.
- $e' := Null$: the MAC value received in the current session, supposedly from Alice.
- $M' := Null$: the message received in the current session, supposedly from Alice.
- a one-trit flag, to distinguish the program states **B0**, **B1**, and **B2**.

Alice and Bob start in program states **A0** and **B0**. We write commit-message(M, i_A) to indicate that Alice is committing herself to sending the message M to Bob in session i_A. We let T be the maximum amount of time Alice waits to receive a response from Bob, and vice versa.

A0 is executed as follows:

```
If $i_A \leq 0$ then **Abort**.
Receive input $(M)$ and commit-message$(M, i_A)$.
Compute $e_{i_A} := \text{MAC}_{a_{i_A}}(i_A \| M)$.
```

```
Send $[e_{i_A}, M]$ to Bob and **goto A1**.
```

B0 is executed as follows:

```
If $i_B \leq 0$ then **Abort**.
Wait to receive $[e', M']$, then **goto B1**.
```

B1 has the following description:

```
Send $[i_B, b_{i_B}]$ to Alice and **goto B2**.
```

A1 is performed in the following manner:

```
Wait at most time $T$ to receive $[i'_B, b']$.
If $[i'_B, b']$ is received, then
    If $i'_B = i_{acceptA}$ and $b_A = b'$ (Bob has not received the last flow of the previous session) then
        Let $N := Null$.
        Send $[i_{acceptA}, a_{i_{acceptA}}, N]$ and **goto A0**.
    If $i'_B = i_A$ and $b_A = H(b')$ then (Alice and Bob seem to be synchronized.)
        Let $N := M$.
        Send $[i_A, a_{i_A}, N]$ to Bob.
        Let $i_{acceptA} := i'_B$, $b_A := b'$ and $i_A := i_A - 1$. (Alice updates her state.)
        **goto A0**.
    else Resend $[e_{i_A}, M]$ to Bob and **goto A1**.
If timeout then
    Resend $[e_{i_A}, M]$ to Bob and **goto A1**.
```

B2 is performed as follows:

```
Wait at most time $T$ to receive $[i'_A, a', N']$.
If $[i'_A, a', N']$ is received, then
```

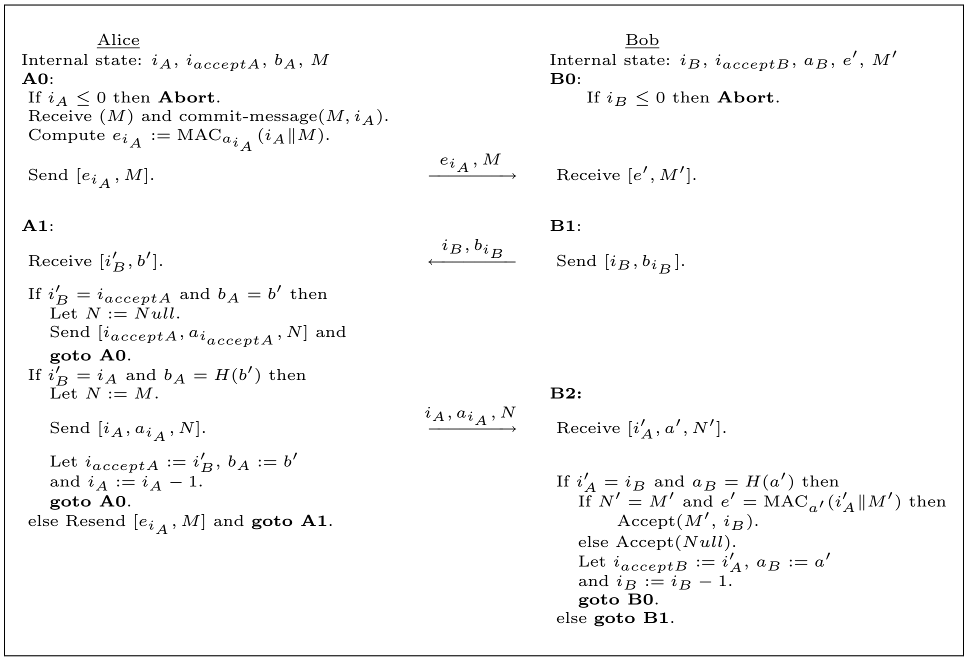
```
    If $i'_A = i_B$ and $a_B = H(a')$ then (Alice and Bob seem to be synchronized.)
        If $N' = M'$ and $e' = \mathrm{MAC}_{a'}(i'_A \| M')$ then
            Accept($M'$, $i_B$).
        else Accept($Null$).
        Let $i_{acceptB} := i'_A$, $a_B := a'$ and $i_B := i_B - 1$. (Bob updates his state.)
        goto B0.
    else goto B1.
If timeout, then goto B1.
```

Figure 4.10 illustrates the main steps of this protocol. For simplicity, the instructions on what to do in case one party does not receive any response from the other party are not included in the figure.

If either Alice or Bob receives a message that they did not expect, they are going to ignore it. For instance, while Alice is in state **A1** and is waiting to receive a message of the form (i_B, b), she is going to ignore messages of the form (M') that request for a new session and correspond to state **A0**. Analogously, when Bob is in state **B2**, he is waiting for a message of type i_A, a, N. He is going to ignore messages of the form e'_{i_A}, M' since they correspond to state **B0**. In general, each party only acts on received messages that have the expected structure in accordance to their current program state.

When Alice is waiting in state **A1** for Bob to respond, she is set to wait for time T. If she receives a message i'_B, b' in time T, then she process it in state **A1**, and otherwise, she resends e_{i_A}, M to Bob. Similarly, Bob waits to receive a message i'_A, a', N', supposedly from Alice, for time T. If he does not receive such a message, he resends i_B, b to Alice.

Note that Eve can block the last flow of Alice, i_A, a, N. In this case, Alice has decremented her state, while Bob is waiting to receive i_A, a, N, and possibly resending i_B, b_{i_B} to remind Alice to send him i_A, a, N. However, since Alice has moved her state to **A0**, she will ignore Bob's messages. This may appear to be problematic since Bob is waiting for Alice. However, once Alice is ready to authenticate a new message to Bob, she will be in program state **A1** again, and communication will resume.



```
Alice
Internal state: $i_A$, $i_{acceptA}$, $b_A$, $M$
A0:
 If $i_A \leq 0$ then Abort.
 Receive ($M$) and commit-message($M, i_A$).
 Compute $e_{i_A} := \text{MAC}_{a_{i_A}}(i_A \| M)$.
 Send [$e_{i_A}, M$].                          $\xrightarrow{e_{i_A}, M}$
A1:
 Receive [$i'_B, b'$].                          $\xleftarrow{i_B, b_{i_B}}$
 If $i'_B = i_{acceptA}$ and $b_A = b'$ then
    Let $N := Null$.
    Send [$i_{acceptA}, a_{i_{acceptA}}, N$] and
    goto A0.
 If $i'_B = i_A$ and $b_A = H(b')$ then
    Let $N := M$.
    Send [$i_A, a_{i_A}, N$].                   $\xrightarrow{i_A, a_{i_A}, N}$
    Let $i_{acceptA} := i'_B$, $b_A := b'$
    and $i_A := i_A - 1$.
    goto A0.
 else Resend [$e_{i_A}, M$] and goto A1.

Bob
Internal state: $i_B$, $i_{acceptB}$, $a_B$, $e'$, $M'$
B0:
    If $i_B \leq 0$ then Abort.
 Receive [$e', M'$].
B1:
 Send [$i_B, b_{i_B}$].
B2:
 Receive [$i'_A, a', N'$].
 If $i'_A = i_B$ and $a_B = H(a')$ then
    If $N' = M'$ and $e' = \text{MAC}_{a'}(i'_A \| M')$ then
        Accept($M'$, $i_B$).
    else Accept($Null$).
    Let $i_{acceptB} := i'_A$, $a_B := a'$
    and $i_B := i_B - 1$.
    goto B0.
 else goto B1.
```

Figure 4.10: Our Proposed Message Recognition Protocol (Common Case)

4.4.1 Security of Our New Message Recognition Protocol

In this section, we consider different types of possible attacks against our protocol. Then, we conclude with a theorem which ensures the security of our protocol.

4.4.1.1 Single-session Attacks

In this section, we consider attacks that are started and completed in a single session. We assume that Eve has stayed passive all along and she becomes active in the current session for the first time. In case of a successful attack, Bob will accept some message M' in the same session, where M' is not $Null$ and not the message sent by Alice in that session. Since Eve has been passive before this session, we will have $i_A = i_B$ at the start of the session; we let $i := i_A = i_B$ for ease of reference. For the same reason, we have $i_{acceptA} = i_{acceptB} = i+1$. Furthermore, Alice and Bob will have accepted all the intended keys so far. That is, $a_B = a_{i+1}$ and $b_A = b_{i+1}$.

We now want to exhaustively list all possible single-session attacks. As in Section 3.2, we follow the notation of [Geh98] in referring to different orderings of the flows. In each attack, the adversary sends a flow to either Alice or Bob and receives a flow in response. This notation labels a flow by **A** if the recipient is Alice, or by **B** when the recipient is Bob. For instance, the following attack scenario corresponds to the attack type of ABAB:

- **A**: Eve sends M to Alice and she responds with e_{i_A}, M.
- **B**: Eve sends e', M' to Bob and he replies with i_B, b_{i_B}.
- **A**: Eve sends i'_B, b' to Alice and receives i_A, a_{i_A}, N from her.
- **B**: Eve sends i'_A, a', N' to Bob.

Recall from Section 3.3.3 that the number of distinct attacks against a three flow protocol is proved to be $\binom{4}{2} = 6$ in [Geh98]. These attacks are denoted AABB, ABBA, BABA, ABAB, BBAA, and BAAB. We will look at these different attacks separately.

One can show that the BABA attack scenario can be reduced to the ABBA attack. That is, if an adversary Oscar can mount a successful attack of type BABA, then Eve can use Oscar and succeed in the ABBA attack scenario. Similarly, we can show that the BAAB and ABBA attack scenarios are reduced to the ABAB case. It remains to analyze the other three attack scenarios, namely AABB, BBAA, and ABAB. We will reduce a successful adversary in these attacks to a player who can mount a depth-i existential forgery or can find depth-i preimages or depth-i second preimages.

Attack of Type AABB Figure 4.11 depicts an attack of type AABB.

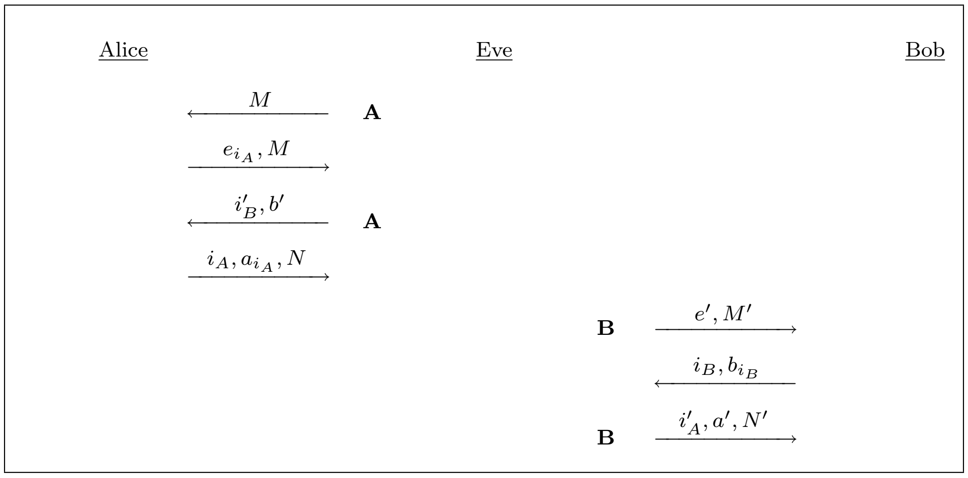


Figure 4.11: Attack of Type AABB

If $i'_A \neq i_B$, Bob will not accept any messages. Since $i_A = i_B = i$, Eve has to set $i'_A := i_A$ in order to succeed. Moreover, Alice reveals i_A and a_{i_A} only if b' is verified; that is, if $b_A = H(b')$ (note that $b_A = b_{i+1}$, as discussed before).

Eve first interacts with Alice and has to find b' before seeing $b_{i_B} = b_i$. This implies that she has found a preimage of $b_A = b_{i+1}$. This exactly translates to the notion of i-PR defined in Def. 6.

Attack of Type BBAA Figure 4.12 illustrates the attack of type BBAA.

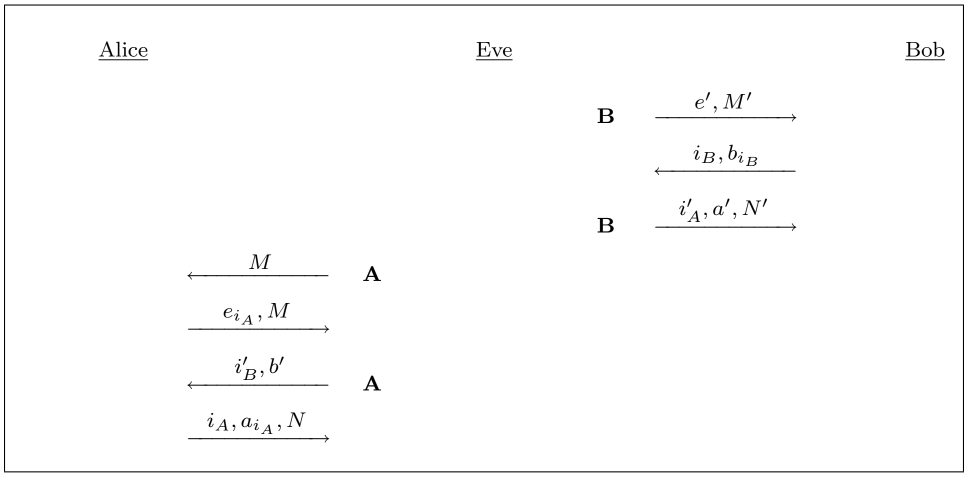


Figure 4.12: Attack of Type BBAA

Alice tries to deceive Bob before she starts interacting with Alice. In order to succeed, Eve needs to present Bob with an a' such that $a_B = H(a')$, without having seen $a_{i_A} = a_i$ (note that $a_B = a_{i+1}$, as discussed before). In other words, she is trying to find a preimage of $a_B = a_{i+1}$. If Eve can successfully find such a preimage, the she translates to a successful player who finds depth-i preimages, as defined in Def. 6.

Attack of Type ABAB Depicted in Fig. 4.13 is the ABAB attack.

In this scenario, Eve receives $b_{i_B} = b_i$ before she has to send b' to Alice. We analyze the two cases $b' = b_i$ and $b' \neq b_i$ separately.

If $b' \neq b_i$, then it implies that Eve has found a depth-i second preimage of $b_A = b_{i+1}$.

Otherwise, $b' = b_i$. Alice will verify $b' = b_i$ and reveal $a_{i_A} = a_i$. Eve now has two choices. She chooses a' such that either $a' = a_{i_A}$ or $a' \neq a_{i_A}$. If $a' \neq a_{i_A}$, then she has found a depth-i second preimage of $a_{i+1} = a_B$. On other hand, if $a' = a_{i_A}$, then for Eve to succeed, she must set $N' := M'$ and she must have set $e' := \mathrm{MAC}_{a'}(i'_A \| M')$ *before* learning a'. That is, Eve has successfully forged a MAC. This reduces to the notion of depth-i existential forgery defined in Def. 8.

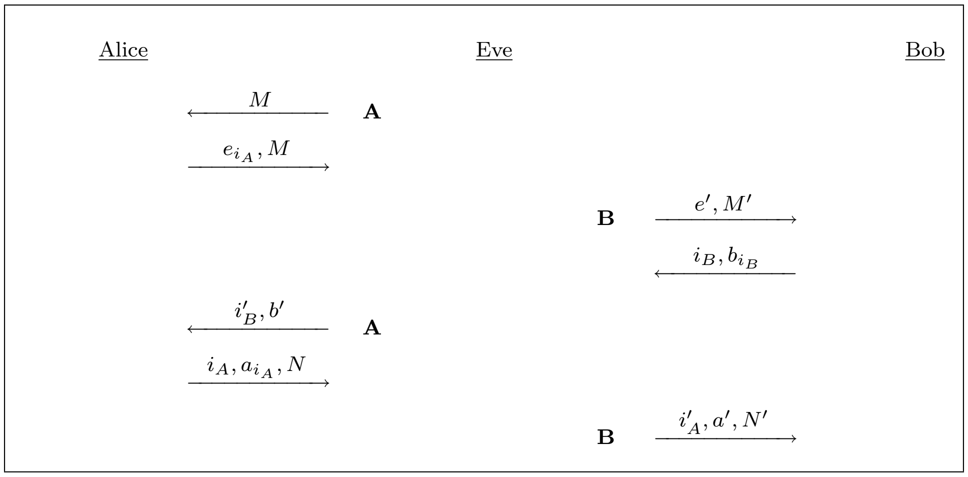


Figure 4.13: Attack of Type ABAB

As was described in Section 3.2 and, also, earlier in this section, Gehrmann [Geh98] formally proves that there are only six possible types of single-session attack against the protocol of Figure 4.10. Here we examine the remaining three attacks: BABA, BAAB, and ABBA. The BABA attack is reduced to the ABBA attack. Then, the ABBA attack is reduced to the ABAB attack. Finally, the BAAB attack is also reduced to the ABAB attack. This concludes the analysis of the six different attack scenarios.

Reducing the BABA attack to an ABBA attack The ABBA attack scenario, depicted in Fig. 4.14, is as follows:

- **A**: Oscar sends M to Alice and receives e_{i_A}, M from her.
- **B**: Oscar sends e', M' to Bob and he sends i_B, b_{i_B}.
- **B**: Oscar sends i'_A, a', N' to Bob.
- **A**: Oscar sends i'_B, b' to Alice and she replies with i_A, a_{i_A}, N.

On the other hand, the BABA attack scenario, illustrated in Fig. 4.15, is as follows:

- **B**: Oscar sends e', M' to Bob and he sends i_B, b_{i_B}.
- **A**: Oscar sends M to Alice and receives e_{i_A}, M from her.
- **B**: Oscar sends i'_A, a', N' to Bob.

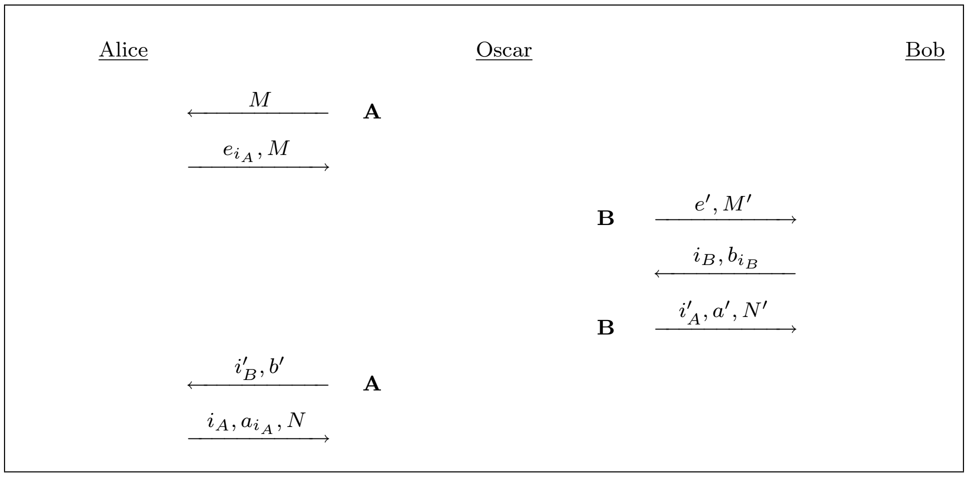


Figure 4.14: Attack of Type ABBA

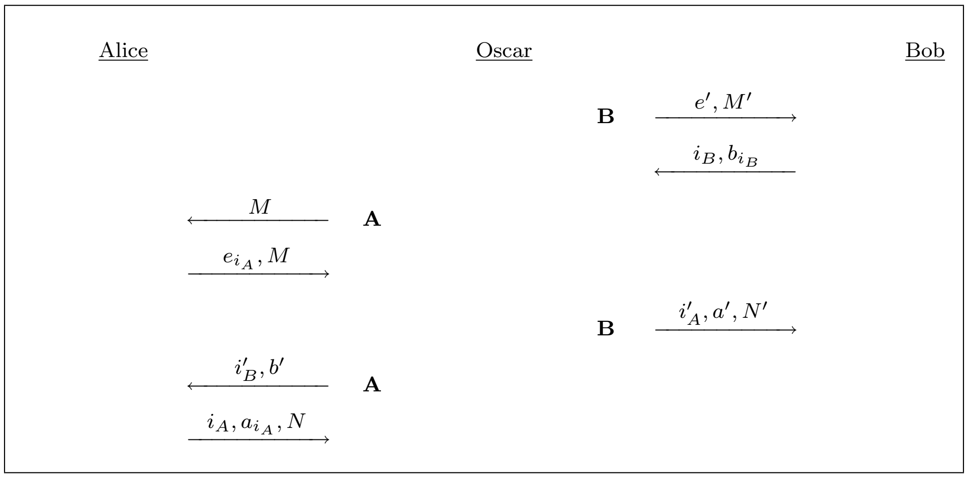


Figure 4.15: Attack of Type BABA

- **A**: Oscar sends i'_B, b' to Alice and she replies with i_A, a_{i_A}, N.

These two attack scenarios differ in the order of the first two steps and are identical otherwise. In the BABA attack scenario, Oscar commits to e' and M' before receiving e_{i_A}. Note that knowing e_{i_A} could possibly help him in choosing e'. On the other hand, Oscar receives i_B and b_{i_B} before sending M. The adversary knows the value of i_B. Moreover, the choice of M is independent of the value of b_{i_B}. In other words, knowing b_{i_B} is not going to help the adversary in choosing M. Hence, if Oscar can win in the BABA attack scenario by first committing to e' and M' and then receiving e_{i_A}, then he can win the ABBA attack scenario with the same values M, M', and e.

Reducing the ABBA attack to an ABAB attack Recall the ABAB attack scenario from Section 4.4.1:

- **A**: Oscar sends M to Alice and receives e_{i_A}, M from her.
- **B**: Oscar sends e', M' to Bob and he sends i_B, b_{i_B}.
- **A**: Oscar sends i'_B, b' to Alice and she replies with i_A, a_{i_A}, N.
- **B**: Oscar sends i'_A, a', N' to Bob.

The ABBA attack differs from the ABAB attack in the order of the last two steps. In the ABAB attack, Oscar receives i_A, a_{i_A}, N from Alice, and then he has to send i'_A, a', N' to Bob. Knowing i_A, a_{i_A}, N can help him choose a winning i'_A, a', N', whereas in the ABBA attack scenario, Oscar sends i'_A, a', N' before seeing i_A, a_{i_A}, N. If Oscar has a winning strategy in the ABBA attack scenario, then using the same values of i'_A, a', N', he will win the ABAB attack scenario.

Reducing the BAAB attack to an ABAB attack The BAAB attack scenario is as follows:

- **B**: Oscar sends e', M' to Bob and he sends i_B, b_{i_B}.
- **A**: Oscar sends M to Alice and receives e_{i_A}, M from her.
- **A**: Oscar sends i'_B, b' to Alice and she replies with i_A, a_{i_A}, N.
- **B**: Oscar sends i'_A, a', N' to Bob.

Figure 4.16 depicts this attack.

The analysis of this case is analogous to that of Section 4.4.1.1. The BAAB attack scenario differs from the ABAB attack scenario in the order of the first two steps. In the BAAB attack scenario, Oscar has to commit to e' and M' before seeing e_{i_A}. Although Oscar receives i_B and b_{i_B} before sending M, these values are independent of the choice of M. That is, seeing b_{i_B} is not going to help the adversary in choosing M. Hence, a winning strategy in the BAAB attack scenario reduces to a winning strategy in the ABAB attack scenario.

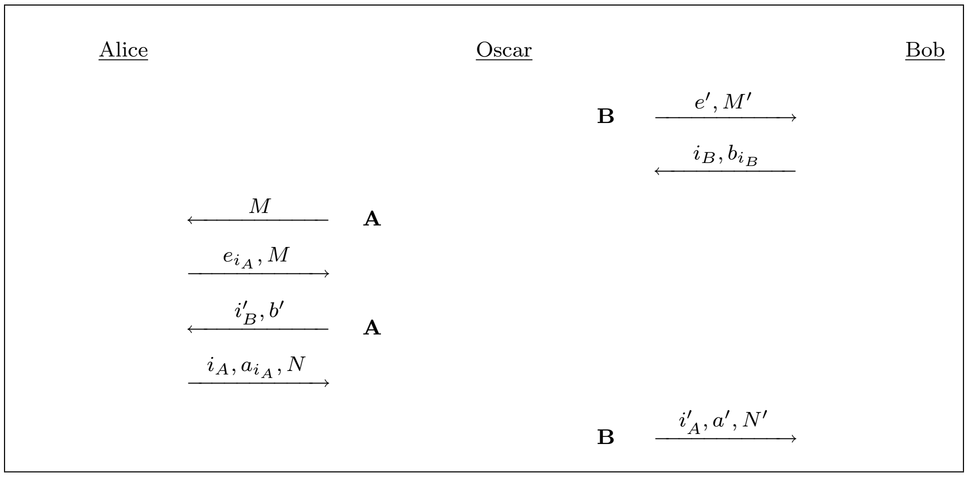


Figure 4.16: Attack of Type BAAB

4.4.1.2 Multi-session Attacks

Having ruled out the possibility of single-session attacks, we now turn our attention to multi-session attacks. Consider attack scenarios which occur over two or more sessions. In such a case, the adversary becomes active in one session and concludes her attack in one of the following sessions. In case of a successful attack, Bob will accept M' in the last session of the attack, where M' is not $Null$ and not the message sent by Alice in that session.

Just before Eve becomes active, similar to the single-session attack scenario discussed above, we must have $i_A = i_B$ and $i_{acceptA} = i_{acceptB} = i_A + 1$. We again let $i := i_A = i_B$ for ease of reference. Moreover, all of the intended keys will have been accepted to this point, so as a result, $a_B = a_{i+1}$ and $b_A = b_{i+1}$.

We now assume that during session i, Eve becomes active by initiating a flow with either Alice or Bob, or changing the information sent by them. Since we are considering multi-session attacks, the attack should not entirely take place in one session. As a result, Eve is not making Bob accept her message M' immediately after she becomes active. The following three cases can happen once Eve becomes active:

Case 1. Bob is not engaged right away. That is, Eve first interacts with Alice.

Case 2. Bob is engaged right away and he outputs the message M, sent by Alice.

Case 3. Bob is engaged right away and he outputs $Null$.

We discuss each case separately.

Case 1. Let us assume that Eve first interacts with Alice and does not engage Bob. In order for Alice to conclude her session, she must receive i'_B, b' such that $i'_B = i$ and $b_{i+1} = H(b')$. Otherwise, Alice will detect that something is going on, hence, she will not reveal i, a_i and, instead, will resend e_i, M. If Eve wants to remain undetected and be able to continue with her attack, she needs to send i'_B, b' such that $i'_B = i$ and $b_{i+1} = H(b')$. This means that Eve has found a depth-i preimage of b_{i+1}.

Case 2. Now assume that Bob is engaged and he outputs the message M, sent by Alice. That is, on input (M), Alice has sent e_i, M to Bob. Since Bob accepts M at the end, it means that he, indeed, has received M in the first flow. Moreover, for Bob to accept M, he must receive i'_A, a', N' such that $i'_A = i$, $a_{i+1} = H(a')$, and $N' = M$. There are three different cases to consider here.

- Not having received i, a_i, M from Alice, Eve finds i'_A, a', N' such that $i'_A = i$ and $a_{i+1} = H(a')$. That is, she finds a depth-i preimage of a_{i+1}.
- Having received i, a_i, M from Alice, Eve finds i'_A, a', N' such that $i'_A = i$, $a_{i+1} = H(a')$, and $a_i \neq a'$. That is, she finds a depth-i second preimage of a_{i+1}.
- Eve sets $i'_A, a', N' = i, a_i, M$. That is, Eve relays Alice's last flow. Note that Alice reveals her last flow only if she receives i'_B, b' such that $i'_B = i$ and $b_{i+1} = H(b')$. There are again three cases to consider here. Either Eve has found a depth-i preimage of b_{i+1}, she has found a depth-i second preimage of b_{i+1}, or she has relayed i, b faithfully. In the latter case, Eve has faithfully relayed all messages, and this does not constitute an attack by an active adversary. This contradicts our assumption that Eve first becomes active in session i.

Case 3. Bob is engaged right away and he outputs $Null$. This means that he has received and verified i'_A and a'. There are again three cases to consider. Either Eve has found a depth-i preimage of a_{i+1}, or she has found a depth-i second preimage of a_{i+1}, or i'_A and a' are the correct i, a_i as revealed by Alice. In this last case, Alice and Bob have successfully remained synchronized, but were unable to authenticate the messages they intended to authenticate.

The above discussion concludes that in the session immediately after Eve becomes active, she can only stop Alice and Bob from authenticating the intended message, but she cannot bring them out of their synchronized states unless she is able to solve the depth-i PR or depth-i SPR problems defined in Definitions 6 and 7. Moreover, if Alice and Bob are synchronized at the beginning of a session, then they will end the session in a synchronized state, unless Eve is able to find depth-i preimages or depth-i second preimages.

At the beginning of a multi-session attack, Alice and Bob are synchronized. The above discussion implies that they remain synchronized until the very last session of the attack. We can look at this last session of the attack separately and think of it as a single-session attack. As a result, any multi-session attack translates to a single-session attack, which were already ruled out in Section 4.4.1.1.

Note that the adversary can only exhaust Alice's and Bob's values of the hash chain one at a time. That is, she can not make them jump more than one step down the hash chain values.

4.4.1.3 Self-recoverability

In this section, we show that once Eve stops interfering with their message flows, Alice and Bob will be able to resume successful communication of recognized messages. Because we have already shown that Alice and Bob remain synchronized in their i values throughout an active attack by Eve (under the security assumptions on H and MAC), we need only show that they do not get "trapped" in a program state, as was the case in the Lucks protocol, for example.

We consider the possible combinations of program states which Alice and Bob are in when Eve becomes passive. We first consider the case where Alice is in state **A1**.

- If Alice is in **A1** and Bob is in **B0**, then after time T, Alice will resend $[e_{i_A}, M]$ to Bob, which will cause him to leave state **B0**, and the protocol will continue.
- If Alice is in **A1** and Bob is in **B1**, then Bob will send $[i_B, b_{i_B}]$ to Alice and advance to **B2**, which will cause her to send an appropriate message to Bob, and herself return to **A0**. Bob will return to **B0**, though he may Accept($Null$) if Eve forged the M' which caused Bob to enter the **B1** state. This can of course only affect the first Accept after Eve's interference, however.
- If Alice is in **A1** and Bob is in **B2**, then Alice will be resending useless messages to Bob, and staying in **A1**, but after time T, Bob will return to **B1**, and we proceed as above.

If Alice is in **A0**, then no progress will be made until the next time she tries to send a message to Bob. At that point, Alice will enter state **A1**, and the analysis continues as above.

4.4.1.4 Main Theorem

The above discussion concludes the discussion of the security and self-recoverability of the proposed message recognition protocol, and forms the proof of the following theorem.

Security and Self-recoverability Theorem. *A successful adversary against the protocol of Section 4.4 who efficiently deceives Bob into accepting (M',i), where M' is not Null and Alice did not send M' in session i, implies an efficient algorithm that finds depth-i preimages or depth-i second preimages, or creates depth-i existential forgeries. Moreover, the adversary cannot stop Alice and Bob from successfully executing the protocol unless she is actively disrupting the communication for the lifetime of Alice and Bob.*

4.5 A New MRP Suitable for Ad Hoc Pervasive Networks

In this section, we describe the details of a protocol which does not employ any hash chain technique. The results presented in this section are drawn from the paper [MS08b]. The initialization phase, execution of the protocol, and the resynchronization process are separately described. The section is concluded by examining the advantages of using this protocol in comparison to previous designs.

We begin by describing the internal states of Alice and Bob. The internal state of Alice includes:

- x_0 and x_1: the *passwords* for this session and the next session, respectively.
- $X_0 = H(x_0)$ and $X_1 = H(x_1)$: the *committing hash values* of the passwords.
- $\mathcal{X}_0 = H(x_0, X_1) = H(x_0, H(x_1))$: the *binding hash value* of the passwords.
- y^*_{-1}, Y^*_0, $\mathcal{Y}^*_0$: Bob's most recent password, committing hash value, and binding hash value accepted by Alice.

Similarly, the internal state of Bob includes:

- y_0 and y_1: the *passwords* for this session and the next session, respectively.
- $Y_0 = H(y_0)$ and $Y_1 = H(y_1)$: the *committing hash values* of the passwords.
- $\mathcal{Y}_0 = H(y_0, Y_1) = H(y_0, H(y_1))$: the *binding hash value* of the passwords.
- x^*_{-1}, X^*_0, $\mathcal{X}^*_0$: Alice's most recent password, committing hash value, and binding hash value accepted by Bob.

In this protocol, x_0 and y_0 are considered to be passwords of the current session. Similarly, x_1 and y_1 are the passwords of the next session. We *commit* to a password by sending its hash value, so that Eve cannot change it. Further, we *bind* two consecutive passwords, in order to detect adversarial intrusions and to be able to resynchronize in such a case.

Alice performs the initialization phase as follows:

Choose random x_0 and x_1.

Compute $X_0 := H(x_0), X_1 := H(x_1)$, and $\mathcal{X}_0 := H(x_0, X_1)$.

Send $X_0, \mathcal{X}_0$ to Bob over the authenticated channel.

Receive $Y_0, \mathcal{Y}_0$ from Bob over the authenticated channel.

Let $y^*_{-1} := \perp, Y^*_0 = Y_0$, and $\mathcal{Y}^*_0 = \mathcal{Y}_0$.

Similarly, Bob executes the initialization phase according to the following steps:

Choose random y_0 and y_1.

Compute $Y_0 := H(y_0), Y_1 := H(y_1)$, and $\mathcal{Y}_0 := H(y_0, Y_1)$.

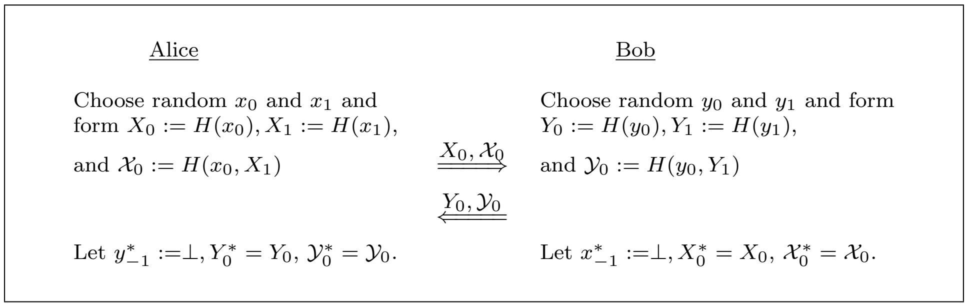
Receive $X_0, \mathcal{X}_0$ from Alice over the authenticated channel.

Send $Y_0, \mathcal{Y}_0$ to Alice over the authenticated channel.

Let $x^*_{-1} := \perp, X^*_0 = X_0$, and $\mathcal{X}^*_0 = \mathcal{X}_0$.

The initialization phase of the protocol is depicted in Fig. 4.17. Next, we move on to the description of the proposed message recognition protocol illustrated in Fig. 4.18.

On input (m, Bob), Alice's execution can be described as follows:



Alice		Bob
Choose random x_0 and x_1 and form $X_0 := H(x_0), X_1 := H(x_1)$, and $\mathcal{X}_0 := H(x_0, X_1)$	$\xrightarrow{X_0, \mathcal{X}_0}$	Choose random y_0 and y_1 and form $Y_0 := H(y_0), Y_1 := H(y_1)$, and $\mathcal{Y}_0 := H(y_0, Y_1)$
	$\xleftarrow{Y_0, \mathcal{Y}_0}$	
Let $y^*_{-1} :=\perp, Y^*_0 = Y_0$, $\mathcal{Y}^*_0 = \mathcal{Y}_0$.		Let $x^*_{-1} :=\perp, X^*_0 = X_0$, $\mathcal{X}^*_0 = \mathcal{X}_0$.

Figure 4.17: Initialization Phase of the New Message Recognition Protocol

Choose a random x_2.

Compute $X_2 := H(x_2), \mathcal{X}_1 := H(x_1, X_2)$, and $d = \mathrm{MAC}_{x_0}[m]$.

Send m, d to Bob and wait to receive $y'_0, Y'_1, \mathcal{Y}'_1$ from Bob. Resend m, d if Bob did not respond.

If $H(y'_0) = Y^*_0$ and $H(y'_0, Y'_1) = \mathcal{Y}^*_0$, then send $(x_0, X_1, \mathcal{X}_1)$ to Bob and update internal state: $y^*_{-1} := y'_0$, $Y^*_0 := Y'_1$, $\mathcal{Y}^*_0 := \mathcal{Y}'_1$, $x_0 := x_1$, $x_1 := x_2$, $X_0 := X_1$, $X_1 := X_2$, and $\mathcal{X}_0 := \mathcal{X}_1$. Otherwise, initiate resynchronization with Bob.

Bob, on the other hand executes the protocol in the following manner:

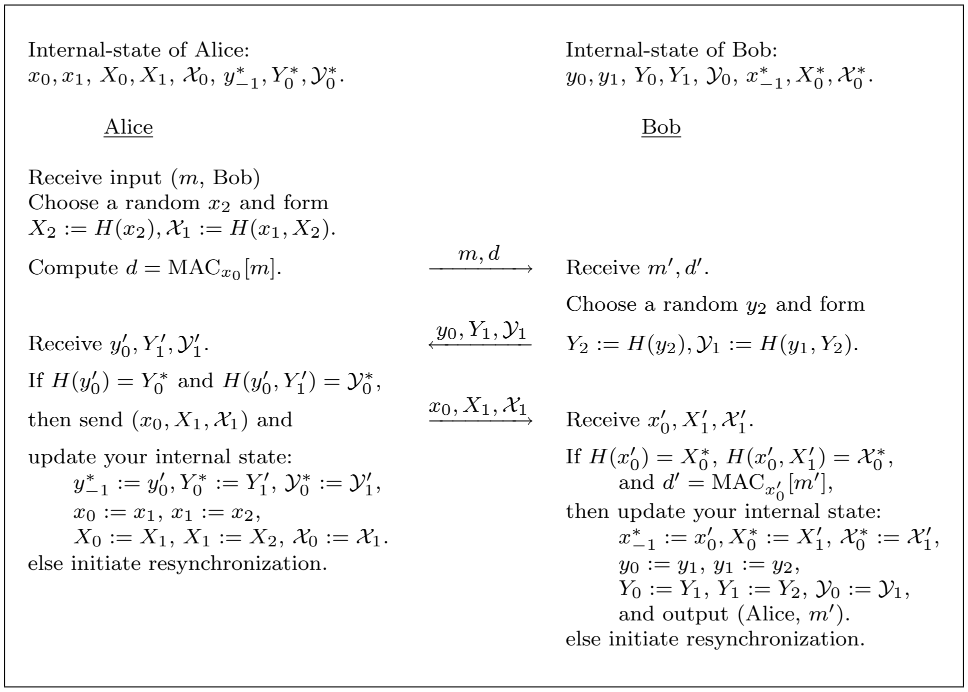
After receiving m', d', choose a random y_2.

Compute $Y_2 := H(y_2)$ and $\mathcal{Y}_1 := H(y_1, Y_2)$.

Send $y_0, Y_1, \mathcal{Y}_1$ to Alice and wait to receive $x'_0, X'_1, \mathcal{X}'_1$. Resend $y_0, Y_1, \mathcal{Y}_1$ to Alice if Alice did not respond.

If $H(x'_0) = X^*_0$ and $H(x'_0, X'_1) = \mathcal{X}^*_0$, and $d' = \mathrm{MAC}_{x'_0}[m']$, then update internal state: $x^*_{-1} := x'_0$, $X^*_0 := X'_1$, $\mathcal{X}^*_0 := \mathcal{X}'_1$, $y_0 := y_1$, $y_1 := y_2$, $Y_0 := Y_1$, $Y_1 := Y_2$, and $\mathcal{Y}_0 := \mathcal{Y}_1$, and output (Alice, m'). Otherwise, initiate resynchronization with Alice.

In case of no adversarial intrusion or communication failure, all the conditions verify and Alice and Bob will not initiate a resynchronization process. When they realize that one of the conditions does not hold, they suspect a communication failure or a possible adversarial intrusion. Hence, they need to resynchronize in order to make sure they have the correct commitment and binding hash values. The synchronization process is illustrated in Fig. 4.19. Bob sends $y_0, Y_1, \mathcal{Y}_1$ to Alice



Internal-state of Alice:
$x_0, x_1, X_0, X_1, \mathcal{X}_0, y^*_{-1}, Y^*_0, \mathcal{Y}^*_0$.

Internal-state of Bob:
$y_0, y_1, Y_0, Y_1, \mathcal{Y}_0, x^*_{-1}, X^*_0, \mathcal{X}^*_0$.

Alice

Receive input (m, Bob)
Choose a random x_2 and form
$X_2 := H(x_2), \mathcal{X}_1 := H(x_1, X_2)$.
Compute $d = \mathrm{MAC}_{x_0}[m]$.

$\xrightarrow{m, d}$

Bob

Receive m', d'.
Choose a random y_2 and form
$Y_2 := H(y_2), \mathcal{Y}_1 := H(y_1, Y_2)$.

$\xleftarrow{y_0, Y_1, \mathcal{Y}_1}$

Alice

Receive $y'_0, Y'_1, \mathcal{Y}'_1$.
If $H(y'_0) = Y^*_0$ and $H(y'_0, Y'_1) = \mathcal{Y}^*_0$,
then send $(x_0, X_1, \mathcal{X}_1)$ and
update your internal state:
$y^*_{-1} := y'_0, Y^*_0 := Y'_1, \mathcal{Y}^*_0 := \mathcal{Y}'_1$,
$x_0 := x_1, x_1 := x_2$,
$X_0 := X_1, X_1 := X_2, \mathcal{X}_0 := \mathcal{X}_1$.
else initiate resynchronization.

$\xrightarrow{x_0, X_1, \mathcal{X}_1}$

Bob

Receive $x'_0, X'_1, \mathcal{X}'_1$.
If $H(x'_0) = X^*_0$, $H(x'_0, X'_1) = \mathcal{X}^*_0$,
and $d' = \mathrm{MAC}_{x'_0}[m']$,
then update your internal state:
$x^*_{-1} := x'_0, X^*_0 := X'_1, \mathcal{X}^*_0 := \mathcal{X}'_1$,
$y_0 := y_1, y_1 := y_2$,
$Y_0 := Y_1, Y_1 := Y_2, \mathcal{Y}_0 := \mathcal{Y}_1$,
and output (Alice, m').
else initiate resynchronization.

Figure 4.18: New Message Recognition Protocol

and Alice sends $x_0, X_1, \mathcal{X}_1$ to Bob. Note that Alice should already have y_0, Y_1 and she is verifying if they match with what she has. Similarly, Bob is verifying if x_0, X_1 match with what he has. However, the values of $\mathcal{X}_1$ and $\mathcal{Y}_1$ are new. It is possible for the adversary to make either Alice or Bob compute a binding hash value in a bogus session. In that case, the binding hash value is refreshed. Note that the resynchronization process is not symmetrical. This is due to the fact that Bob may detect an intrusion after Alice has updated her state. In this case, the values $x_0, X_1, \mathcal{X}_1$ that Alice sends during the resynchronization process need to be verified differently.

Since we are not using a hash chain, the memory requirement on the devices is relaxed. The octuple $(x_0, x_1, X_0, X_1, \mathcal{X}_0, y^*_{-1}, Y^*_0, \mathcal{Y}^*_0)$ is all Alice needs to communicate with Bob (she will need another octuple for each different user). In the previous protocols, the devices had to deal with a hash chain for every single device they wanted to communicate with. Storing all the values of a hash chain, for example $a_0, a_1, \ldots, a_n$, is too demanding for low-end devices. On the other hand, storing only the root value of the hash chain, for instance a_0, requires too many computations at each session. The alternative is to employ a time-storage trade-off and store some of the hash values, see, for example, [CJ03]. Still, there are some storage and computational requirements associated with this implementation. Our proposal for not having to deal with a hash chaining technique avoids any memory

Internal-state of Alice:
$x_0, x_1,\ X_0, X_1,\ \mathcal{X}_0,\ y^*_{-1}, Y^*_0, \mathcal{Y}^*_0$.

Internal-state of Bob:
$y_0, y_1,\ Y_0, Y_1,\ \mathcal{Y},\ x^*_{-1}, X^*_0, \mathcal{X}^*_0$.

<u>Alice</u>

Choose a random x_2 and form
$X_2 := H(x_2), \mathcal{X}_1 := H(x_1, X_2)$.

Receive $y'_0, Y'_1, \mathcal{Y}'_1$.

If $y^*_{-1} = y'_0$ and $Y^*_0 = Y'_1$,
then $\mathcal{Y}^*_0 := \mathcal{Y}'_1$,
else initiate resynchronization.

$\xleftarrow{y_0, Y_1, \mathcal{Y}_1}$

$\xrightarrow{x_0, X_1, \mathcal{X}_1}$

<u>Bob</u>

Choose a random y_2 and form
$Y_2 := H(y_2), \mathcal{Y}_1 := H(y_1, Y_2)$.

Receive $x'_0, X'_1, \mathcal{X}'_1$.

If $x^*_{-1} = x'_0$ and $X^*_0 = X'_1$,
then $\mathcal{X}^*_0 := \mathcal{X}'_1$,
else if $H(x'_0) = X^*_0$ and $H(x'_0, X'_1) = \mathcal{X}^*_0$,
then $x^*_{-1} := x'_0, X^*_0 := X'_1, \mathcal{X}^*_0 := \mathcal{X}'_1$.
else initiate resynchronization.

Figure 4.19: The Resynchronization Process

or computational requirement of this nature for every session.

Moreover, the passwords are set to be chosen at random in each session. Hence, they are independent of one another and are refreshed in each session. As a result, we do not need to consider assumptions that depend on the number of sessions the protocol is executed. Consequently, the security does not weaken as the protocol is executed over time.

Furthermore, the devices can run this protocol as many times as they want and the total number of sessions is not fixed. This provides extra flexibility compared to the protocols based on the hash chain technique. Next, we look at the security assumptions relevant for this new protocol.

4.5.1 Security Assumptions

In this section, we define new notions of hash function security, namely **Paired Preimage Resistance (PPR)**, **Paired Second Preimage Resistance (PSPR)**, **Binding Unforgeability (BU)**, and **Binding Preimage Resistance (BPR)**. Each notion is presented as a game between a player Oscar and a Challenger. Note that these assumptions are independent of the number of times the protocol has been executed. In other words, in contrast to the approach taken by Lucks et al. [LZWW05], where they have to assume "depth-i non-invertibility", "depth-i second preimage resistance", "depth-i unforgeability", and "depth-i combined security" for every i, $1 \leq i \leq n$, we only require four assumptions.

The PPR notion is depicted in Fig. 4.20. We note that the PPR property is analogous to the notion of "depth-2 non-invertibility" defined by Lucks et al.

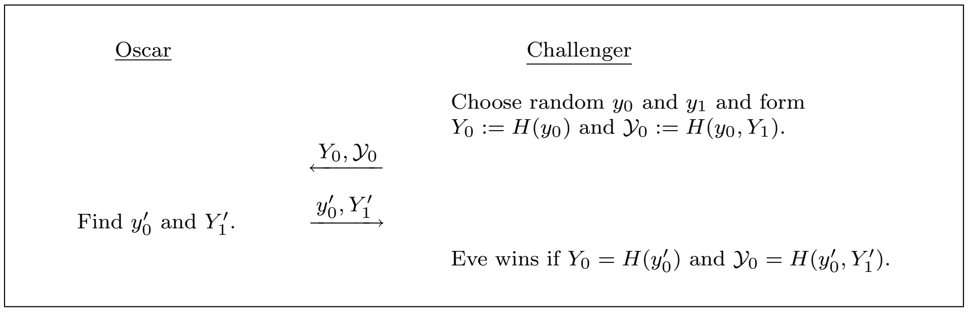

Oscar
Challenger
Choose random y_0 and y_1 and form $Y_0 := H(y_0)$ and $\mathcal{Y}_0 := H(y_0, Y_1)$.
$\xleftarrow{Y_0, \mathcal{Y}_0}$
Find y'_0 and Y'_1.
$\xrightarrow{y'_0, Y'_1}$
Eve wins if $Y_0 = H(y'_0)$ and $\mathcal{Y}_0 = H(y'_0, Y'_1)$.

Figure 4.20: Paired Preimage Resistance

[LZWW05]. Furthermore, this one assumption is replacing a whole family of assumptions, termed "depth-i non-invertibility", for $1 \leq i \leq n$.

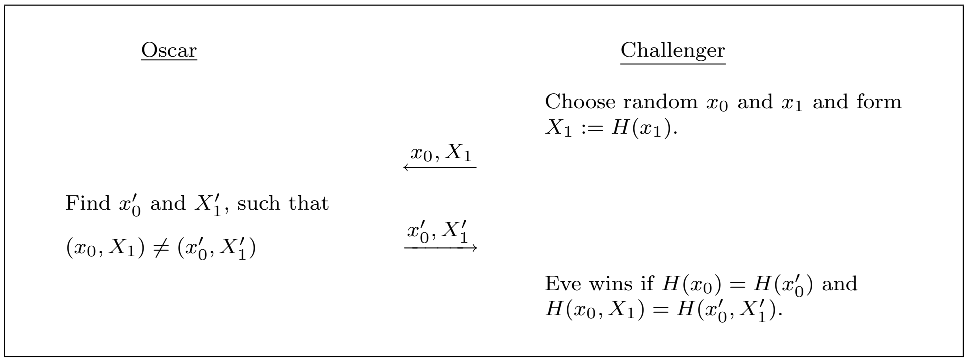

Oscar
Challenger
Choose random x_0 and x_1 and form $X_1 := H(x_1)$.
$\xleftarrow{x_0, X_1}$
Find x'_0 and X'_1, such that $(x_0, X_1) \neq (x'_0, X'_1)$
$\xrightarrow{x'_0, X'_1}$
Eve wins if $H(x_0) = H(x'_0)$ and $H(x_0, X_1) = H(x'_0, X'_1)$.

Figure 4.21: Paired Second-preimage Resistance

Figure 4.21 illustrates the PSPR notion. This notion is analogous to "depth-2 second preimage resistance" defined by Lucks et al. [LZWW05]. It is replacing the family of assumptions termed "depth-i second preimage resistance", for i, $1 \leq i \leq n$.

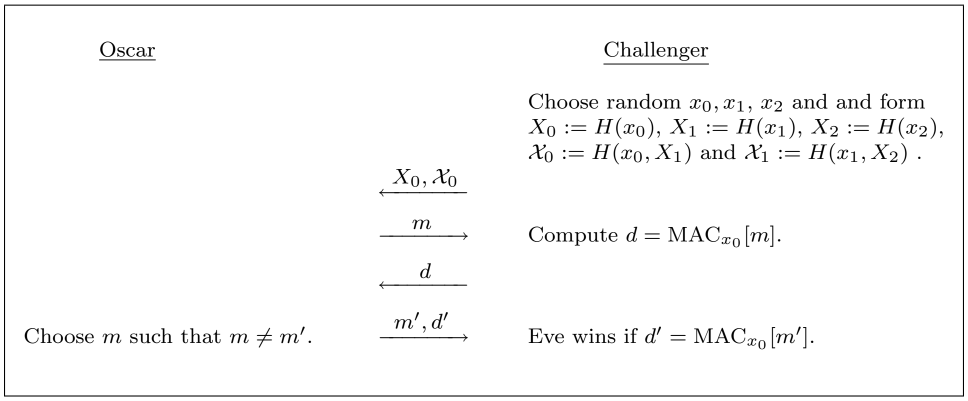

Oscar
Challenger
Choose random x_0, x_1, x_2 and and form $X_0 := H(x_0)$, $X_1 := H(x_1)$, $X_2 := H(x_2)$, $\mathcal{X}_0 := H(x_0, X_1)$ and $\mathcal{X}_1 := H(x_1, X_2)$.
$\xleftarrow{X_0, \mathcal{X}_0}$
$\xrightarrow{m}$
Compute $d = \mathrm{MAC}_{x_0}[m]$.
$\xleftarrow{d}$
Choose m such that $m \neq m'$.
$\xrightarrow{m', d'}$
Eve wins if $d' = \mathrm{MAC}_{x_0}[m']$.

Figure 4.22: Binding Unforgeability

The notion of BU is depicted in Fig. 4.22. Analogous to this notion, Lucks et al.

[LZWW05] define "depth-2 unforgeability". Note that the BU notion is replacing a family of assumptions termed "depth-i unforgeability", for i, $1 \leq i \leq n$.

In Section 4.5.2, we will see that the BU, PPR, and PSPR notions prevent attacks that start and finish during one session. Moreover, attack scenarios spanning over two sessions are also analyzed, and the BPR notion illustrated in Fig. 4.23 is associated with these attacks.

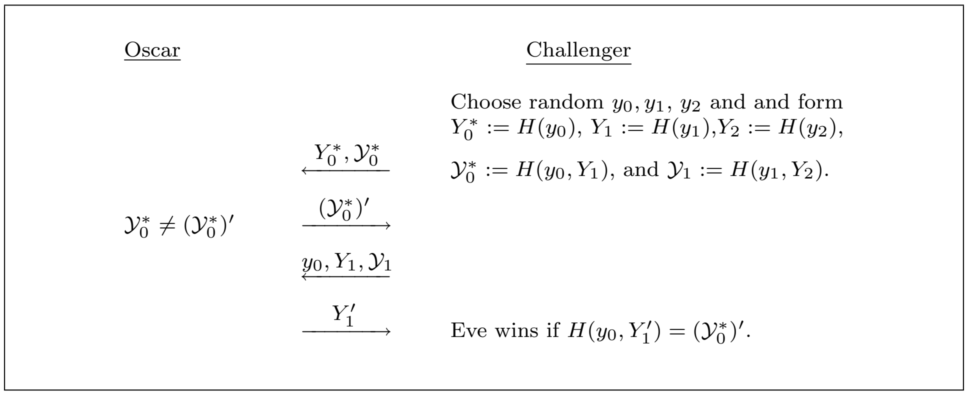

Oscar
Challenger
Choose random y_0, y_1, y_2 and and form $Y_0^* := H(y_0)$, $Y_1 := H(y_1)$,$Y_2 := H(y_2)$,
$\xleftarrow{Y_0^*, \mathcal{Y}_0^*}$
$\mathcal{Y}_0^* := H(y_0, Y_1)$, and $\mathcal{Y}_1 := H(y_1, Y_2)$.
$\mathcal{Y}_0^* \neq (\mathcal{Y}_0^*)'$
$\xrightarrow{(\mathcal{Y}_0^*)'}$
$\xleftarrow{y_0, Y_1, \mathcal{Y}_1}$
$\xrightarrow{Y_1'}$
Eve wins if $H(y_0, Y_1') = (\mathcal{Y}_0^*)'$.

Figure 4.23: Binding Preimage Resistance

Next, we prove the security of our protocol, based on the assumption that the PPR, PSPR, BU, and BPR games are hard to win.

4.5.2 Security of the Proposed Recognition Protocol

Recall that the goal of the adversary is to make Bob accept a message m' that was never sent from Alice. A successful attack is where that Bob is deceived and he outputs (Alice, m').

Let $(x_0, x_1, X_0, X_1, \mathcal{X}_0, y_{-1}^*.Y_0^*, \mathcal{Y}_0^*)$ and $(y_0, y_1, Y_0, Y_1, \mathcal{Y}_0, x_{-1}^*, X_0^*, \mathcal{X}_0^*)$ be the internal states of Alice and Bob, respectively. Now, assume that Eve, having been passive all along, mounts a successful attack for the first time and Bob actually outputs (Alice, m'), where $m \neq m'$. Since, Eve had been passive before this attack, we can assume that $y_{-1}^* = y_0, Y_0^* = H(y_0) = Y_0, \mathcal{Y}_0^* = \mathcal{Y}_0 = H(y_0, H(y_1))$, $x_{-1}^* = x_0, X_0^* = X_0 = H(x_0)$, and $\mathcal{X}_0^* = \mathcal{X}_0 = H(x_0, H(x_1))$. Eve may complete her attack in one session, or she may mount an attack that spans more than one session. First, we examine one-round attacks.

4.5.2.1 One-session Attacks

In order to exhaustively list all possible one-round attacks against our protocol, similar to Sections 3.4 and 4.4.1.1, we use the notation of Gehrmann [Geh98] in labelling different orderings of the flows. Recall, we label each flow sent by the adversary by **A**, if the recipient is Alice, and by **B**, when the recipient is Bob. For example, an ordering of ABAB corresponds to the following attack scenario:

- **A**: Eve sends m to Alice and Alice responds with m, d.
- **B**: Eve sends m', d' to Bob and Bob replies with $y_0, Y_1, \mathcal{Y}_1$.
- **A**: Eve sends $y_0', Y_1', \mathcal{Y}_1'$ to Alice and receives $x_0, X_1, \mathcal{X}_1$ from her.
- **B**: Eve sends $x_0', X_1', \mathcal{X}'_1$.

As it was mentioned before, there are $\binom{4}{2} = 6$ possible attacks, namely AABB, ABBA, BABA, ABAB, BBAA, and BAAB. Next, we will analyze each of these attack scenarios.

We prove that the BABA attack scenario can be reduced to the ABBA attack. In other words, if the adversary can mount a successful attack of type BABA, then she also succeeds in the ABBA attack scenario. Similarly, one can show that the BAAB and ABBA attack scenarios can be reduced to the ABAB case. Hence, it remains to investigate the AABB, BBAA, and ABAB attack scenarios. We prove that the AABB, BBAA, and ABAB attacks are not possible by reducing them to the PPR, PSPR, or PCR games. Then, we show the aforementioned reductions.

Attack of Type AABB. The attack of type AABB is illustrated in Fig. 4.24. In this attack scenario, Eve finishes her interactions with with Alice before she starts her interactions with Bob. In other words, Eve has to first deceive Alice in order to get her to reveal the information she needs to then deceive Bob.

If Eve successfully deceives Alice, then she receives $(x_0, X_1, \mathcal{X}_1)$. Now, Eve computes $d' = \mathrm{MAC}_{x_0}[m']$, for m' of her choice. She then sends m', d' to Bob. Finally, she completes her attack with setting $(x_0', X_1', \mathcal{X}_1') = (x_0, X_1, \mathcal{X}_1)$ and sending it to Bob.

In order to deceive Alice, Eve has to find y_0' and Y_1' such that $Y_0 = H(y_0')$ and $\mathcal{Y}_0 = H(y_0', Y_1')$, where Y_0 and $\mathcal{Y}_0$ were transmitted in the session immediately before the attack. Note that Eve, not having seen (y_0, Y_1), has sent (y_0', Y_1'), which

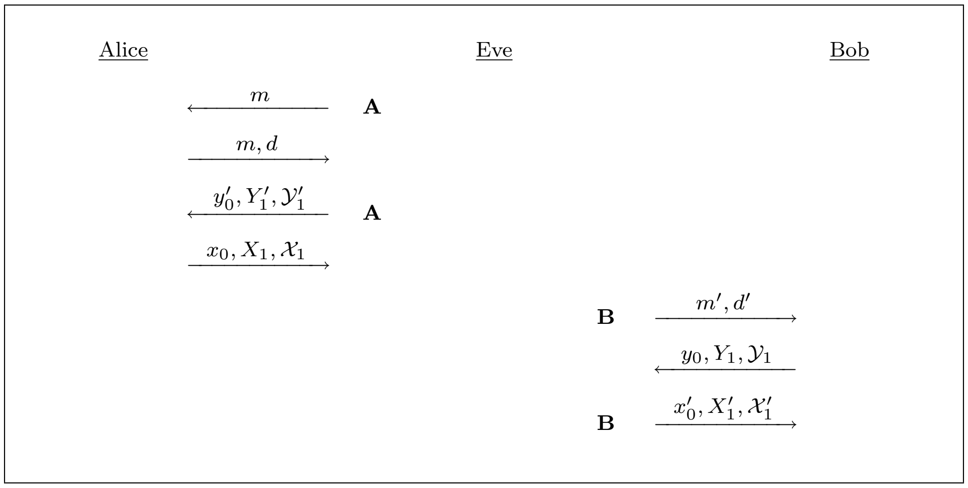

Alice
Eve
Bob
m
A
m, d
$y'_0, Y'_1, \mathcal{Y}'_1$
A
$x_0, X_1, \mathcal{X}_1$
B
m', d'
$y_0, Y_1, \mathcal{Y}_1$
B
$x'_0, X'_1, \mathcal{X}'_1$

Figure 4.24: Attack of Type AABB

has been accepted by Alice. This is exactly the problem of PPR, depicted in Fig. 4.20.

Attack of Type BBAA. The attack of type BBAA is illustrated in Fig. 4.25. In this scenario, Eve interacts with Alice after she has finished interacting with Bob. That is, she receives $(y_0, Y_1, \mathcal{Y}_1)$ from Bob before she has to choose $(y'_0, Y'_1, \mathcal{Y}'_1)$. If she chooses $(y'_0, Y'_1, \mathcal{Y}'_1)$ such that $(y_0, Y_1) \neq (y'_0, Y'_1)$ and remains undetected by Alice, then, Eve can be reduced to a successful player against the PSPR game of Fig. 4.21. We deal with the case where $(y_0, Y_1) = (y'_0, Y'_1)$ and $\mathcal{Y}_1 \neq \mathcal{Y}'_1$ in Section 4.5.2.2. The only remaining case is that, having received $(y_0, Y_1, \mathcal{Y}_1)$ from Bob, Eve lets $(y'_0, Y'_1, \mathcal{Y}'_1) = (y_0, Y_1, \mathcal{Y}_1)$ to avoid being detected by Alice.

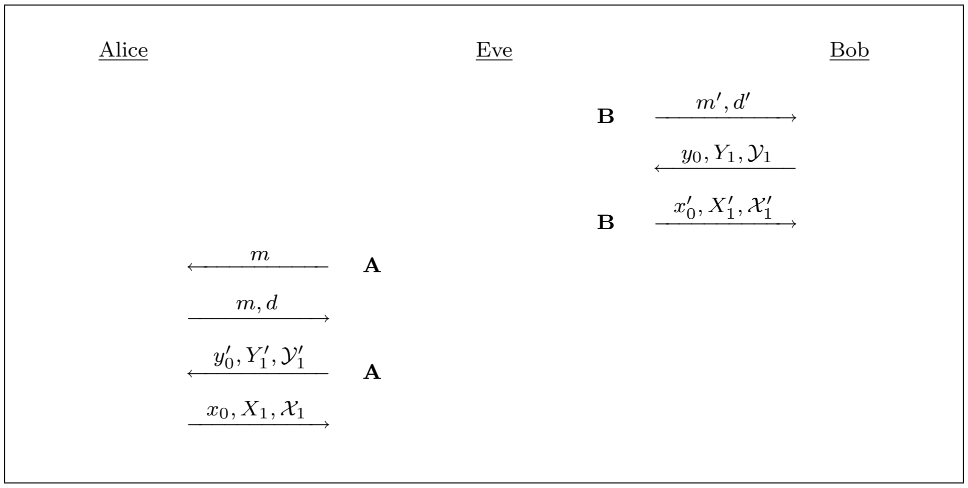

Alice
Eve
Bob
B
m', d'
$y_0, Y_1, \mathcal{Y}_1$
B
$x'_0, X'_1, \mathcal{X}'_1$
m
A
m, d
$y'_0, Y'_1, \mathcal{Y}'_1$
A
$x_0, X_1, \mathcal{X}_1$

Figure 4.25: Attack of Type BBAA

A successful attack of this type implies that Bob has accepted m'. That is, not having seen (x_0, X_1), Eve has found x'_0, X'_1. Once Eve finds the appropriate x'_0 and X'_1, she can compute $d' = \text{MAC}_{x'_0}[m']$, for an m' of her choice. Note that Eve has received X_0 and $\mathcal{X}_0$ from the previous session. Now, she has to find x'_0, X'_1 such that $X_0 = H(x'_0)$ and $\mathcal{X}_0 = H(x'_0, X'_1)$. This translates to the notion of PPR if we replaces each x value by its corresponding y value.

Attack of Type ABAB. Figure 4.26 illustrates the attack of type ABAB. In this attack, Eve receives the correct $(y_0, Y_1, \mathcal{Y}_1)$ from Bob before she has to send $(y'_0, Y'_1, \mathcal{Y}'_1)$ to Alice. As it was discussed in the case of the BBAA attack, Eve will be detected by Alice unless she sets $(y'_0, Y'_1, \mathcal{Y}'_1) = (y_0, Y_1, \mathcal{Y}_1)$. This way Alice will not detect Eve and she will reveal $(x_0, X_1, \mathcal{X}_1)$. The adversary has two choices now. She either sets $(x'_0, X'_1) = (x_0, X_1)$ and send it to Bob, or she sends (x'_0, X'_1) to Bob where $(x'_0, X'_1) \neq (x_0, X_1)$. We will analyze each of these two cases separately.

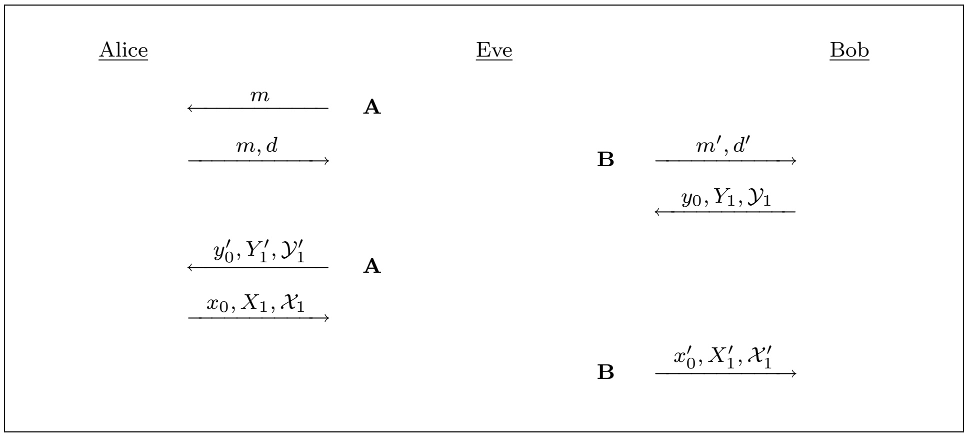

Alice
Eve
Bob
m
A
m, d
B
m', d'
$y_0, Y_1, \mathcal{Y}_1$
$y'_0, Y'_1, \mathcal{Y}'_1$
A
$x_0, X_1, \mathcal{X}_1$
B
$x'_0, X'_1, \mathcal{X}'_1$

Figure 4.26: Attack of Type ABAB

Let us first consider the case where $(x'_0, X'_1) = (x_0, X_1)$. In this case, the adversary has collected $(X_0, \mathcal{X}_0)$ from previous session. She then sends m to Alice and Alice replies with (m, d). She will then send (m', d') to Bob. At this point the rest of the flows are determined to be the following: She receives $(y_0, Y_1, \mathcal{Y}_1)$ from Bob, sets $(y'_0, Y'_1, \mathcal{Y}'_1) = (y_0, Y_1, \mathcal{Y}_1)$, and sends it to Alice. Further, she receives $(x_0, X_1, \mathcal{X}_1)$ from Alice, lets $(x'_0, X'_1, \mathcal{X}'_1) = (x_0, X_1, \mathcal{X}_1)$, and sends it Bob. Hence, this case is exactly the notion of BU depicted in Fig. 4.22.

The second case is when $(x'_0, X'_1) \neq (x_0, X_1)$. Assume that Eve can mount a successful attack of type ABAB with $(x'_0, X'_1) \neq (x_0, X_1)$. That is, she has collected $X_0, \mathcal{X}_0$ from previous session. She chooses m and receives d such that $d = \text{MAC}_{x_0}[m]$. Then, she submits m', d'. Finally, she receives x_0, X_1 and she is supposed to send x'_0, X'_1 such that $(x'_0, X'_1) \neq (x_0, X_1)$, $H(x_0) = H(x'_0)$, $H(x_0, X_1) =$

$H(x'_0, X'_1)$, and $d' = \text{MAC}_{x'_0}[m']$. We reduce Eve to a successful player against the Challenger of PSPR game, depicted in Fig. 4.21. The reduction is illustrated in Fig. 4.27.

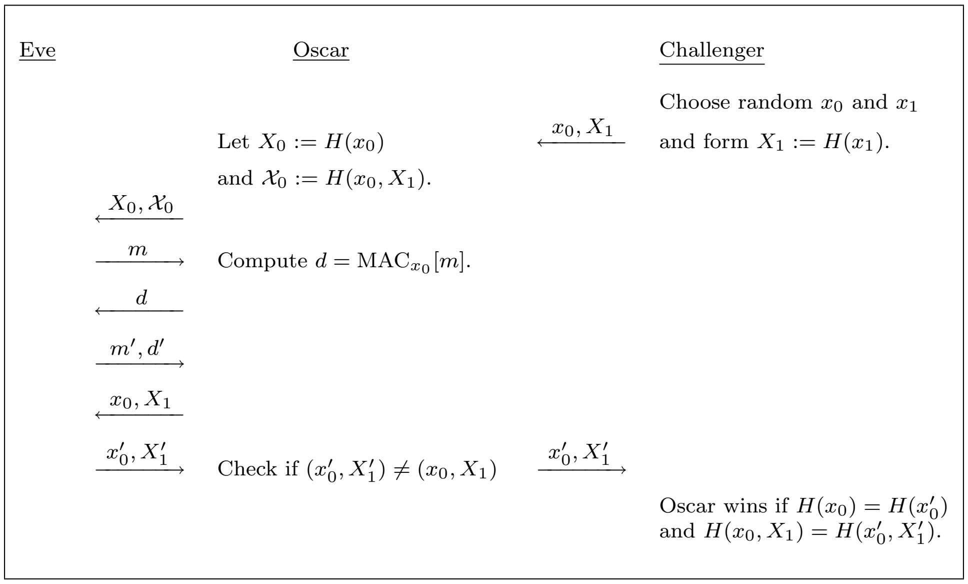

Eve
Oscar
Challenger
Choose random x_0 and x_1 and form $X_1 := H(x_1)$.
x_0, X_1
Let $X_0 := H(x_0)$ and $\mathcal{X}_0 := H(x_0, X_1)$.
$X_0, \mathcal{X}_0$
m
Compute $d = \text{MAC}_{x_0}[m]$.
d
m', d'
x_0, X_1
x'_0, X'_1
Check if $(x'_0, X'_1) \neq (x_0, X_1)$
x'_0, X'_1
Oscar wins if $H(x_0) = H(x'_0)$ and $H(x_0, X_1) = H(x'_0, X'_1)$.

Figure 4.27: Reducing Eve to a Player Against the Challenger of PSPR

Note that, Oscar is playing against the Challenger of PSPR and at the same time he is playing the role of both Alice and Bob against Eve.

We continue by reducing the BABA attack to the ABBA attack. Further, we reduce the ABBA attack to the ABAB attack that was analyzed in Section 4.5.2. Finally, the only remaining attack scenario, BAAB, is also reduced to the ABAB attack. This concludes the analysis of the six different attack scenarios.

Reducing BABA attack to ABBA attack. The attack of type ABBA is depicted in Fig. 4.28, and Fig. 4.29 illustrates the attack of Type BABA.

These two attacks differ only in the order of the first two steps. The ABBA attack is as follows:

- **A**: Oscar sends m to Alice and she responds with m, d.
- **B**: Oscar sends m', d' to Bob and he replies with $y_0, Y_1, \mathcal{Y}_1$.
- **B**: Oscar sends $x'_0, X'_1, \mathcal{X}'_1$.
- **A**: Oscar sends $y'_0, Y'_1, \mathcal{Y}'_1$ to Alice and receives $x_0, X_1, \mathcal{X}_1$ from her.

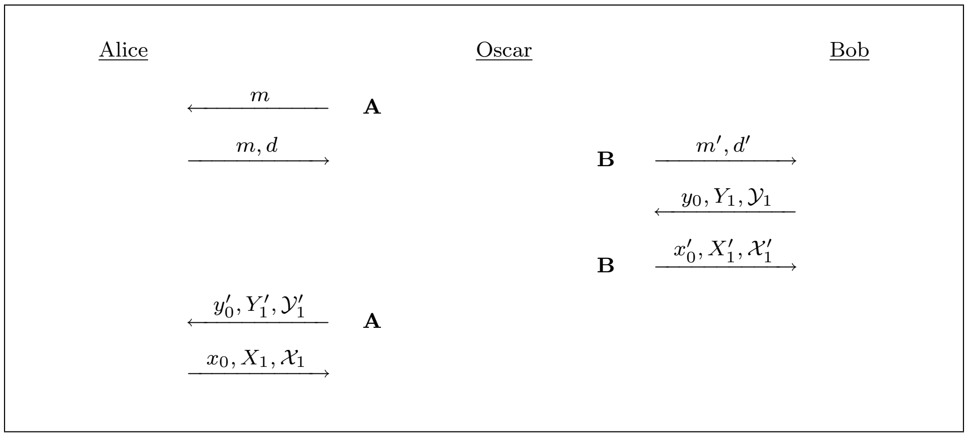

Alice
Oscar
Bob
m
A
m, d
B
m', d'
$y_0, Y_1, \mathcal{Y}_1$
B
$x'_0, X'_1, \mathcal{X}'_1$
$y'_0, Y'_1, \mathcal{Y}'_1$
A
$x_0, X_1, \mathcal{X}_1$

Figure 4.28: Attack of Type ABBA

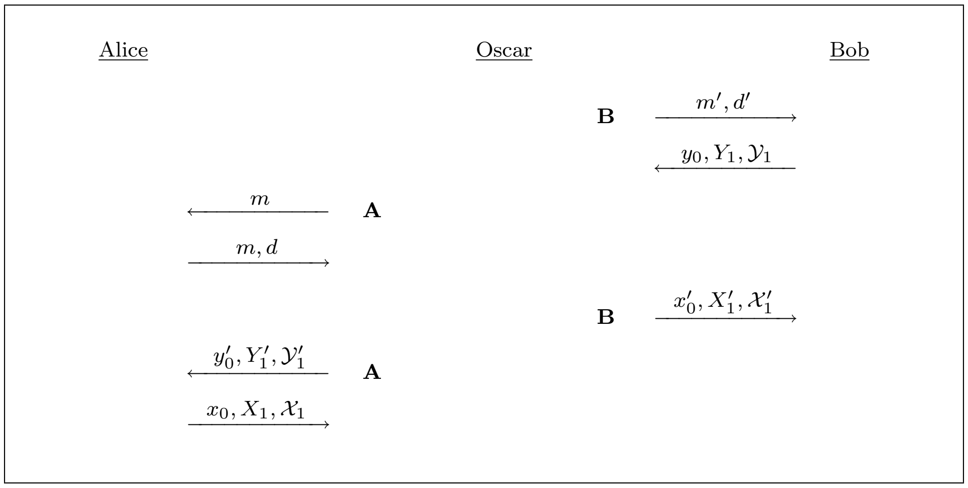

Alice
Oscar
Bob
B
m', d'
$y_0, Y_1, \mathcal{Y}_1$
m
A
m, d
B
$x'_0, X'_1, \mathcal{X}'_1$
$y'_0, Y'_1, \mathcal{Y}'_1$
A
$x_0, X_1, \mathcal{X}_1$

Figure 4.29: Attack of Type BABA

The BABA attack has the following order:

- **B**: Oscar sends m', d' to Bob and he replies with $y_0, Y_1, \mathcal{Y}_1$.
- **A**: Oscar sends m to Alice and she responds with m, d.
- **B**: Oscar sends $x'_0, X'_1, \mathcal{X}'_1$.
- **A**: Oscar sends $y'_0, Y'_1, \mathcal{Y}'_1$ to Alice and receives $x_0, X_1, \mathcal{X}_1$ from her.

Note that in the BABA attack scenario, the choice of m is independent of what the values of y_0, Y_1 and $\mathcal{Y}_1$ are. That is, knowing $y_0, Y_1, \mathcal{Y}_1$ before choosing m is not going to help Oscar. On the other hand, he is committing himself to m', d' before receiving any values, such as d, that could possibly help him. If Oscar wins by first choosing m', d' and then receiving d in the BABA attack scenario, then he can also win the ABBA attack by using the same values m, m', and d'.

Reducing ABBA attack to ABAB. Recall the ABAB attack described in Section 4.5.2:

- **A**: Eve sends m to Alice and she responds with m, d.
- **B**: Eve sends m', d' to Bob and he replies with $y_0, Y_1, \mathcal{Y}_1$.
- **A**: Eve sends $y'_0, Y'_1, \mathcal{Y}'_1$ to Alice and receives $x_0, X_1, \mathcal{X}_1$ from her.
- **B**: Eve sends $x'_0, X'_1, \mathcal{X}'{}_1$.

This attack differers from the ABBA attack in the order of the last two steps. In the ABAB attack, Eve first receives $x_0, X_1, \mathcal{X}_1$ from Alice, then she has to send $x'_0, X'_1, \mathcal{X}'{}_1$ to Bob. Whereas in the case of the ABBA attack, Oscar has to send $x'_0, X'_1, \mathcal{X}'{}_1$ to Bob before he receives $x_0, X_1, \mathcal{X}_1$ from Alice. If Oscar has a winning strategy in the ABBA attack, the Eve can use him in her ABAB attack by sending the same values of $x'_0, X'_1, \mathcal{X}'{}_1$ that Oscar sends to Bob.

Reducing BAAB attack to ABAB. Depicted in Fig. 4.30 is the attack Type of BAAB.

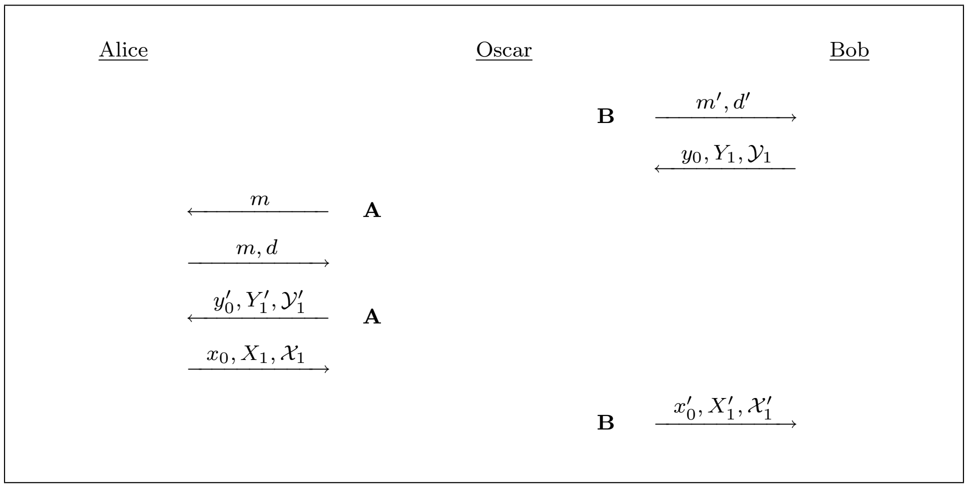

Alice
Oscar
Bob
B
m', d'
$y_0, Y_1, \mathcal{Y}_1$
m
A
m, d
$y'_0, Y'_1, \mathcal{Y}'_1$
A
$x_0, X_1, \mathcal{X}_1$
B
$x'_0, X'_1, \mathcal{X}'_1$

Figure 4.30: Attack of Type BAAB

Recall that knowing $y_0, Y_1, \mathcal{Y}_1$ before choosing m is not going to help Oscar. Moreover, in the BAAB attack, Oscar is first committing himself to m', d'. If Oscar wins the BAAB attack by first choosing m', d' and then receiving d, then so will Eve in the ABAB attack, by just using the same values m, m', and d'.

4.5.2.2 Multi-session Attacks

Now consider attack scenarios which span two or more sessions. The adversary is active but remains undetected in all sessions of the attack. She then submits her message in the last session of the attack. If she tampers with y_0, Y_1, x_0, or X_1 and remains undetected, then we go back to the cases described above. Hence, it remains to examine the cases when she changes the binding hash values. We look at the case where Eve changes the value of $\mathcal{Y}_1$ to $\mathcal{Y}_1'$. The case where Eve alters $\mathcal{X}_1$ to $\mathcal{X}_1'$ is analogous due to the symmetry of the protocol structure.

Assume that Eve changes $\mathcal{Y}_1$ to $\mathcal{Y}_1'$ and does not touch y_0 or Y_1. She goes undetected in this session because Alice verifies y_0 and Y_1, but only records $\mathcal{Y}_1'$ without verification. She then updates her state as follows $y_{-1}^* := y_0$, $Y_0^* := Y_1$, $(\mathcal{Y}_0^*)' := \mathcal{Y}_1'$.

In the next session, Alice sends $(y_0, Y_1, \mathcal{Y}_1)$ and Eve has to change it to an appropriate $(y_0', Y_1', \mathcal{Y}_1')$ to remain undetected. Otherwise, Alice will call for resynchronization. Alice checks to see if $H(y_0') = Y_0^*$ and $H(y_0', Y_1') = (\mathcal{Y}_0^*)'$. We treat the two cases $y_0 = y_0'$ and $y_0 \neq y_0'$ separately.

If $H(y_0') = Y_0^*$ and $y_0 \neq y_0'$, then Eve, having seen y_0, has found y_0' such that $H(y_0') = H(y_0)$. This means that Eve has found a second preimage of y_0.

On the other hand, when $y_0 = y_0'$, the condition $H(y_0') = Y_0^*$ holds. Then, Alice verifies to see if $H(y_0, Y_1') = (\mathcal{Y}_0^*)'$. If it holds, then Eve is a successful player in the BPR game of Fig. 4.23.

If the adversary were to mount an attack that spans over more than two rounds, she would have to successfully pass the second round. However, the above discussion shows that the adversary can only pass the second session without being detected if she can win the BPR game or SPR game.

4.5.2.3 The Security Theorem

We investigated all possible attacks against the message recognition protocol of Fig. 4.18 by considering two different cases, namely if the attack is taking place over one session, or if it spans more than one session. We examined these two cases separately.

In the first case, there are six possible attack scenarios: BABA, BAAB, ABBA, AABB, BBAA, and ABAB. Attacks of type BABA, BAAB, and ABBA can be reduced to the ABAB case. Further, we showed that a successful adversary (Eve) in attacks of type AABB, BBAA, and ABAB attacks can be reduced to a successful player (Oscar) in the PPR, PSPR, or BU games.

In the case of attacks that occur over more than one session, we showed that the successful adversary can be reduced to a successful player against the BPR or SPR games.

This concludes the analysis of different attack scenarios and proves the following theorem.

Theorem 8. *A successful adversary, who can efficiently deceive Bob in outputting (Alice, m'), where Alice never sent m', implies an efficient algorithm in winning PPR, PSPR, BU, or BPR hash function games.*

This theorem precisely identifies the required properties for a hash function to be used in the message recognition protocol of Fig. 4.18. There is no concrete construction of such a hash function. However, no one knows how to prove that a concrete construction of a hash function has any non-trivial property. It is a standard approach taken in the literature to assume some properties for an idealized hash function and to prove security of a given protocol assuming these assumptions. Note that the same approach was taken by Lucks et al. [LZWW05]. One can analyze these games in the random oracle model and compare their hardness to more standard hash function security notions, see for example Sections 2.3.2.

Chapter 5

Conclusion and Future Work

Contents

5.1 A Summary of the Thesis

We assumed that there are two channels available for communication, one insecure broadband channel and one authenticated narrow-band channel. We produced the required formalism needed in a general model of non-interactive Message Authentication Protocols using these two channels. GNIMAP depicts a general non-interactive Message Authentication Protocol. We proved that GNIMAP is secure given that a Binding Game is hard to win for an adversary with certain properties.

Further, we examined the NIMAPs found in the literature. We discussed their security in our general model. We proposed a particular NIMAP based on HCR hash functions. We proved that our proposed NIMAP is secure in the general model given that the HCR Game is hard to win.

Our proposed NIMAP, sends the same amount of information over the authenticated channel as the most secure NIMAP proposed so far, while achieving the same level of security. In comparison with this latter protocol, our NIMAP reduces the amount of information sent over the insecure channel significantly.

Next, having examined the most secure and efficient IMAP found in the literature, we proposed a new IMAP based on ICR hash functions, a new notion that we have defined. Given a secure ICR hash function, we proved that our IMAP is secure.

Our security assumptions are reasonable and are based on the existence of an ICR hash function. We do not require any previously distributed public parameters, which are needed for commitment schemes.

The amount of information sent over the authenticated channel is smaller than the most secure IMAP proposed so far, while achieving the same level of security. Allowing the same amount of information to be sent over the authenticated channel, we can tolerate much stronger adversaries.

Finally, we examined the problem of message recognition. Previous recognition protocols were revisited and their shortcomings were pointed out. We looked at the Lucks protocol in more detail and described a situation where the protocol fails to recover after the adversary's intrusion. We suggested a variant of this protocol to overcome this problem. In particular, in case of communication failure or adversarial disruption, this protocol is not equipped with a practical resynchronization process and can fail to resume. Our proposed variant is equipped with a resynchronization technique that allows users to resynchronize whenever they wish or when they suspect an intrusion. We have also noted the equivalence of digital signature schemes with message recovery and non-interactive message recognition protocols.

We further proposed a new message recognition protocol, which is based on the original protocol by Lucks et al., and which incorporates a resynchronization technique within itself to provide self-recoverability. That is, the proposed protocol overcomes the recoverability problem of the Lucks et al. protocol without having to provide a separate resynchronization procedure. Finally, we formally proved the security of our protocol.

Last but not least, we proposed a new design for message recognition protocols suitable for ad hoc pervasive networks. This proposal does not make use of hash chains. Hash chaining techniques have been used in recent designs of message recognition protocols. In this approach, the small devices are required to save values of a hash chain in their memories for every single user they want to communicate with. Since we do not use this technique, we no longer require the small devices to save values of a hash chain in their memories. This relaxes the memory requirements and makes the protocol more suitable for ad hoc networks.

5.2 Future Work and Outlook

The security of some message authentication protocols is based on computational assumptions, the hardness of ICR and HCR hash function games, for instance. The existence of particular instances of secure ICR and HCR hash functions is an open problem. It is also interesting to propose protocols that are based on other assumptions that are well-studied, such as collision resistance or second-preimage resistance. On another note, we have provided a general framework for NIMAPs, and it would be interesting to see a general model for IMAPs as well. Proposing a model that encapsulates all the possibilities in the interactive setting of IMAPs is going to be a harder task when compared to the case of NIMAPs. There are more possible flow structures in an interactive setting which results in more possible attack scenarios.

A related direction is to investigate other cryptographic goals, for example mutual key generation in the context of ad hoc networks using two-channel cryptography. Mutual key generation happens when both parties contribute random inputs of their choice to a protocol that generates a key. This allows both parties to ensure that the generated key is sufficiently random and hard to predict. It would be interesting to provide some Diffie-Hellman type protocols achieving mutual key generation that take advantage of the narrow-band channel. It is important that these protocols be light-weight in terms of communication bandwidth and memory requirements.

There are also some interesting problems involving the alternate usages of the narrow-band channel, the implementation of this channel, and assuming weaker properties for this channel. In the literature, all applications of the narrow-band channel happen simultaneously with the use of the broadband channel. However, one can think of scenarios where using the authenticated flow beforehand is an advantage. For instance, there are password based schemes for key agreement protocols. A bidirectional authenticated channel can be used to transmit the short password to Alice and Bob. In general, one can see if incorporating the manual channel into any existing protocol will result in some security advantage.

Implementing the narrow-band channel is a significant step in the development of two-channel cryptography. The early suggestions involved incorporating human abilities in designing authentication protocols. However, human beings are prone to error and it is desirable to eliminate the human error factor. Infrared (IR), laser, near field communication (NFC) developed by Sony and Phillips, or visible light between the two devices can be used to send a short string. One can also require the two devices to physically touch each other. Although using these signals has the advantage of essentially eliminating the human error factor, there is a cost associated to equipping the devices with the appropriate signal transmitter and receiver. Many manufacturing companies may be reluctant to equip their devices with such transmitters as a result. An alternative to the latter two proposals is to use hash function chains in implementing the second channel. This approach eliminates the human error and it is very cost efficient since almost all devices are equipped with built-in hash functions anyway.

One other point in realization of the narrow-band channel is that, for most proposals, it is not possible to replay flows over this authenticated channel. The only exception seems to be voice over IP in which the voice can be recorded by the adversary and replayed later on to authenticate a different message. For all other instances, it seems that the adversary cannot replay the flows and hence is bound to change the communication over the wireless channel in realtime. This will pose a weakening of adversarial capabilities and will enable researchers to design protocols which achieve the same level of security and, yet, be more efficient and based on simpler and weaker security assumptions. That is, they have to transmit less information and have lower computational complexity for a comparable level of security.

References

[ABC$^+$98] Ross Anderson, Francesco Bergadano, Bruno Crispo, Jong-Hyeon Lee, Charalampos Manifavas, and Roger Needham. A new family of authentication protocols. In *ACMOSR: ACM Operating Systems Review*, volume 32, pp. 9–20, 1998. 7, 9, 59, 63

[BSSW02] Dirk Balfanz, Diana K. Smetters, Paul Stewart, and H. Chi Wong. Talking to strangers: authentication in ad-hoc wireless networks. In *Network and Distributed System Security Symposium*, San Diego, California, U.S.A., February 2002. 4, 7, 9, 22

[CEEC08] Marie Chan, Daniel Estève, Christophe Escriba, and Eric Campo. A review of smart homes-present state and future challenges. *Computer Methods and Programs in Biomedicine*, 91(1):55–81, July 2008. 3

[CJ03] Don Coppersmith and Markus Jakobsson. Almost optimal hash sequence traversal. In *Financial Cryptography*, *Lecture Notes in Computer Science*, volume 2357, pp. 102–119. Springer, 2003. 90

[Dem04] George Demiris. Electronic home healthcare: concepts and challenges. *International Journal of Electronic Healthcare*, 1(1):4–16, 2004. 4

[Geh98] Christian Gehrmann. Multiround unconditionally secure authentication. *Designs, Codes, and Cryptography*, 15(1):67–86, 1998. 39, 78, 79, 81, 94

[GMN04] Christian Gehrmann, Chris J. Mitchell, and Kaisa Nyberg. Manual authentication for wireless devices. *RSA Cryptobytes*, 7(1):29–37, January 2004. 4, 7, 9, 11, 23

[GMR88] Shafi Goldwasser, Silvio Micali, and Ron L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, 1988. 5

[GMS08] Ian Goldberg, Atefeh Mashatan, and Douglas R. Stinson. A new message recognition protocol with self-recoverability for ad hoc pervasive networks. Technical Report 2008-22, Centre for Applied Cryptographic Research (CACR), University of Waterloo, Canada, 2008. 10, 60, 74

[GN04] Christian Gehrmann and Kaisa Nyberg. Security in personal area networks. *Security for Mobility, IEE, London*, pp. 191–230, 2004. 4, 7, 9, 11, 16

[Hoe04] Jaap-Henk Hoepman. The ephemeral pairing problem. In *Financial Cryptography*, *Lecture Notes in Computer Science*, volume 3110, pp. 212–226. Springer, 2004. 4, 7, 9, 36

[HWGW05] Jonathan Hammell, André Weimerskirch, Joao Girao, and Dirk Westhoff. Recognition in a low-power environment. In *ICDCSW '05: Proceedings of the Second International Workshop on Wireless Ad Hoc Networking (WWAN) ICDCSW'05)*, pp. 933–938, Washington, DC, USA, 2005. IEEE Computer Society. 7, 9, 10, 59, 64

[LAN05] Sven Laur, N. Asokan, and Kaisa Nyberg. Efficient mutual data authentication using manually authenticated strings: Preliminary version. Cryptology ePrint Archive, Report 2005/424, 2005. A shorter version was published at CANS 2006. 4, 7, 9

[LN06] Sven Laur and Kaisa Nyberg. Efficient mutual data authentication using manually authenticated strings. In *The 5th International Conference on Cryptology and Network Security, CANS 2006, Suzhou, Dec. 8 - 10, 2006*, *Lecture Notes in Computer Science*, volume 4301. Springer, 2006. To appear. It is a shortened version of ePrint Report 2005/424. 4, 7, 9

[LZWW05] Stefan Lucks, Erik Zenner, André Weimerskirch, and Dirk Westhoff. Entity recognition for sensor network motes. In *GI Jahrestagung (2)*, pp. 145–149, 2005. 5, 9, 10, 59, 64, 66, 68, 91, 92, 93, 101

[LZWW07] Stefan Lucks, Erik Zenner, André Weimerskirch, and Dirk Westhoff. Is this Message From Alice? Efficient and Secure Entity Recognition for Low-End Devices, 2007. Manuscript. 7, 9, 66

[LZWW08] Stefan Lucks, Erik Zenner, André Weimerskirch, and Dirk Westhoff. Concrete security for entity recognition: The Jane Doe protocol. In *Progress in Cryptology - INDOCRYPT 2008, 9th International Conference on Cryptology in India, Kharagpur, India, December 14-17, 2008. Proceedings*, *Lecture Notes in Computer Science*, volume 5365, pp. 158–171. Springer, 2008. 7, 9, 66, 68

[Mit03] Chris J. Mitchell. Remote user authentication using public information. In *IMA Int. Conf.*, *Lecture Notes in Computer Science*, volume 2898, pp. 360–369. Springer, 2003. 7, 9, 59, 63

[MS07] Atefeh Mashatan and Douglas R. Stinson. Noninteractive two-channel message authentication based on hybrid-collision resistant hash functions. *IET Information Security*, 1(3):111–118, September 2007. 9, 12, 55

[MS08a] Atefeh Mashatan and Douglas R. Stinson. Interactive two-channel message authentication based on interactive-collision resistant hash functions. *To appear in International Journal of Information Security*, 2008. 9, 16, 34

[MS08b] Atefeh Mashatan and Douglas R. Stinson. A new message recognition protocol for ad hoc pervasive networks. 2008. To appear in The 7th International Conference on Cryptology and Network Security (CANS 2008). 10, 60, 87

[MS08c] Atefeh Mashatan and Douglas R. Stinson. Recognition in ad hoc pervasive networks. Technical Report 2008-12, Centre for Applied Cryptographic Research (CACR), University of Waterloo, Canada, 2008. 9, 10, 60

[MvOV96] Alfred Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996. 6

[NSS06] Moni Naor, Gil Segev, and Adam Smith. Tight bounds for unconditional authentication protocols in the manual channel and shared key models., *Lecture Notes in Computer Science*. volume 4117, pp. 214–231. Springer, 2006. 4, 7, 9, 35, 37, 52, 56, 58

[PV06] Sylvain Pasini and Serge Vaudenay. An optimal non-interactive message authentication protocol. In David Pointcheval, editor, *Topics in Cryptography*, *Lecture Notes in Computer Science*, volume 3860, pp. 280–294, San Jose, California, U.S.A., February 2006. Springer. 4, 7, 9, 21, 23, 24, 29, 55

[RS84] Ronald L. Rivest and Adi Shamir. How to expose an eavesdropper. *Communications of the ACM*, 27(4):393–394, 1984. 2

[RWSN07] Mohammad Reza Reyhanitabar, Shuhong Wang, and Reihaneh Safavi-Naini. Non-interactive manual channel message authentication based on etcr hash functions. In *Information Security and Privacy, 12th Australasian Conference, ACISP 2007, Townsville, Australia, July 2-4, 2007, Proceedings*, *Lecture Notes in Computer Science*, volume 4586, pp. 385–399. Springer, 2007. 4, 7, 9

[SA99] Frank Stajano and Ross Anderson. The resurrecting duckling: security issues for ad-hoc wireless networks. In *Security Protocols, Seventh International Workshop Proceedings*, *Lecture Notes in Computer Science*, volume 1796. Springer, 1999. 4, 7, 9

[Vau05] Serge Vaudenay. Secure communications over insecure channels based on short authenticated strings. In *Advances in Cryptography, CRYPTO 05: The 25th Annual International Cryptology Conference*, *Lecture Notes in Computer Science*, volume 3621, pp. 309–326, Santa Barbara, California, U.S.A., August 2005. Springer. 4, 7, 9, 23, 36, 52, 55

[WSN08] Shuhong Wang and Reihaneh Safavi-Naini. New results on unconditionally secure multireceiver manual authentication. Cryptology ePrint Archive, Report 2008/039, 2008. 4, 7, 9, 12, 30, 31

[WW03] André Weimerskirch and Dirk Westhoff. Zero common-knowledge authentication for pervasive networks. In *Selected Areas in Cryptography*, *Lecture Notes in Computer Science*, volume 3006, pp. 73–87. Springer, 2003. 6, 7, 9, 10, 59, 64